MapReduce Service

Component Operation Guide

Issue 01

Date 2024-12-13

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 Using CarbonData

1.1 CarbonData Data Types

Description

In CarbonData, data is stored in entities called tables. CarbonData tables are similar to RDBMS tables which organize data in rows and columns. CarbonData tables store structured data, and have fixed columns and data types.

Supported Data Types

CarbonData tables support the following data types:

- Int
- String
- BigInt
- Smallint
- Char
- Varchar
- Boolean
- Decimal
- Double
- TimeStamp
- Date
- Array
- Struct
- Map

The following table describes supported data types and their respective values ranges.

Table 1-1 CarbonData data types

Data Type	Value Range
Int	4-byte signed integer ranging from -2,147,483,648 to 2,147,483,647. **NOTE** **int** data in a non-dictionary column is internally stored as the **BigInt** type.
String	100,000 characters **NOTE** If the **CHAR** or **VARCHAR** data type is used in **CREATE TABLE**, the two data types are automatically converted to the String data type. If a column contains more than 32,000 characters, add the column to the **LONG_STRING_COLUMNS** attribute of the **tblproperties** table during table creation.
BigInt	64-bit value ranging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
SmallInt	–32,768 to 32,767
Char	A to Z and a to z
Varchar	A to Z, a to z, and 0 to 9
Boolean	**true** or **false**
Decimal	The default value is (10,0) and maximum value is (38,38). **NOTE** When query with filters, append **BD** to the number to achieve accurate results. For example, **select * from carbon_table where num = 1234567890123456.22BD**.
Double	64-bit value ranging from 4.9E-324 to 1.7976931348623157E308
TimeStamp	The default format is **yyyy-MM-dd HH:mm:ss**.
Date	The **DATE** data type is used to store calendar dates. The default format is **yyyy-MM-DD**.
Array<data_type>	N/A **NOTE** Currently, only two layers of complex types can be nested.
Struct<col_name: data_type COMMENT col_comment, ...>	
Map<primitive_type, data_type>	

Main Specifications of CarbonData

Table 1-2 Main specifications

Entity	Tested Value	Test Environment
Number of tables	10000	3 nodes. 4 vCPUs and 20 GB memory for each executor. Driver memory: 5 GB, 3 executors. Total columns: 107 String: 75 Int: 13 BigInt: 7 Timestamp: 6 Double: 6
Number of table columns	2000	3 nodes. 4 vCPUs and 20 GB memory for each executor. Driver memory: 5 GB, 3 executors.
Maximum size of a raw CSV file	200GB	17 cluster nodes. 150 GB memory and 25 vCPUs for each executor. Driver memory: 10 GB, 17 executors.
Number of CSV files in each folder	100 folders. Each folder has 10 files. The size of each file is 50 MB.	3 nodes. 4 vCPUs and 20 GB memory for each executor. Driver memory: 5 GB, 3 executors.
Number of load folders	10000	3 nodes. 4 vCPUs and 20 GB memory for each executor. Driver memory: 5 GB, 3 executors.

Table Specifications

Table 1-3 Table specifications

Entity	Tested Value
Number of secondary index tables	10
Number of composite columns in a secondary index table	5

Entity	Tested Value
Length of column name in a secondary index table (unit: character)	120
Length of a secondary index table name (unit: character)	120
Cumulative length of all secondary index table names + column names in an index table* (unit: character)	3800**

NOTE

- * Characters of column names in an index table refer to the upper limit allowed by Hive or the upper limit of available resources.
- ** Secondary index tables are registered using Hive and stored in HiveSERDEPROPERTIES in JSON format. The value of **SERDEPROPERTIES** supported by Hive can contain a maximum of 4,000 characters and cannot be changed.

1.2 CarbonData Table User Permissions

The following table provides details about Hive ACL permissions required for performing operations on CarbonData tables.

Prerequisites

Parameters listed in **Table 1-21** or **Table 1-22** have been configured.

Hive ACL permissions

Table 1-4 Hive ACL permissions required for CarbonData table-level operations

Scenario	Required Permission
DESCRIBE TABLE	SELECT (of table)
SELECT	SELECT (of table)
EXPLAIN	SELECT (of table)
CREATE TABLE	CREATE (of database)
CREATE TABLE As SELECT	CREATE (on database), INSERT (on table), RW on data file, and SELECT (on table)
LOAD	INSERT (of table) RW on data file
DROP TABLE	OWNER (of table)
DELETE SEGMENTS	DELETE (of table)
SHOW SEGMENTS	SELECT (of table)

Scenario	Required Permission
CLEAN FILES	DELETE (of table)
INSERT OVERWRITE / INSERT INTO	INSERT (of table) RW on data file and SELECT (of table)
CREATE INDEX	OWNER (of table)
DROP INDEX	OWNER (of table)
SHOW INDEXES	SELECT (of table)
ALTER TABLE ADD COLUMN	OWNER (of table)
ALTER TABLE DROP COLUMN	OWNER (of table)
ALTER TABLE CHANGE DATATYPE	OWNER (of table)
ALTER TABLE RENAME	OWNER (of table)
ALTER TABLE COMPACTION	INSERT (on table)
FINISH STREAMING	OWNER (of table)
ALTER TABLE SET STREAMING PROPERTIES	OWNER (of table)
ALTER TABLE SET TABLE PROPERTIES	OWNER (of table)
UPDATE CARBON TABLE	UPDATE (of table)
DELETE RECORDS	DELETE (of table)
REFRESH TABLE	OWNER (of main table)
REGISTER INDEX TABLE	OWNER (of table)
SHOW PARTITIONS	SELECT (on table)
ALTER TABLE ADD PARTITION	OWNER (of table)
ALTER TABLE DROP PARTITION	OWNER (of table)

NOTE

- If tables in the database are created by multiple users, the **Drop database** command fails to be executed even if the user who runs the command is the owner of the database.
- In a secondary index, when the parent table is triggered, **insert** and **compaction** are triggered on the index table. If you select a query that has a filter condition that matches index table columns, you should provide selection permissions for the parent table and index table.
- The LockFiles folder and lock files created in the LockFiles folder will have full permissions, as the LockFiles folder does not contain any sensitive data.
- If you are using ACL, ensure you do not configure any path for DDL or DML which is being used by other process. You are advised to create new paths.

  Configure the path for the following configuration items:

  1) carbon.badRecords.location

  2) Db_Path and other items during database creation
- For Carbon ACL in a non-security cluster, **hive.server2.enable.doAs** in the **hive-site.xml** file must be set to **false**. Then the query will run as the user who runs the hiveserver2 process.

1.3 Creating a CarbonData Table Using the Spark Client

This section describes how to create CarbonData tables, load data, and query data. This quick start provides operations based on the Spark Beeline client. If you want to use Spark shell, wrap the queries with **spark.sql()**.

The following describes how to load data from a CSV file to a CarbonData table.

Table 1-5 CarbonData Quick Start

Operation	Description
Preparing a CSV File	Prepare the CSV file to be loaded to the CarbonData Table.
Connecting to CarbonData	Connect to CarbonData before performing any operations on CarbonData.
Creating a CarbonData Table	Create a CarbonData table to load data and perform query operations.
Loading Data to a CarbonData Table	Load data from CSV to the created table.
Querying Data from a CarbonData Table	Perform query operations such as filters and groupby.

Preparing a CSV File

1. Prepare a CSV file named **test.csv** on the local PC. An example is as follows:

   13418592122,1001, *MAC address*, 2017-10-23 15:32:30,2017-10-24 15:32:30,62.50,74.56
   13418592123 1002, *MAC address*, 2017-10-23 16:32:30,2017-10-24 16:32:30,17.80,76.28
   13418592124,1003, *MAC address*, 2017-10-23 17:32:30,2017-10-24 17:32:30,20.40,92.94
   13418592125 1004, *MAC address*, 2017-10-23 18:32:30,2017-10-24 18:32:30,73.84,8.58
   13418592126,1005, *MAC address*, 2017-10-23 19:32:30,2017-10-24 19:32:30,80.50,88.02
   13418592127 1006, *MAC address*, 2017-10-23 20:32:30,2017-10-24 20:32:30,65.77,71.24
   13418592128,1007, *MAC address*, 2017-10-23 21:32:30,2017-10-24 21:32:30,75.21,76.04
   13418592129,1008, *MAC address*, 2017-10-23 22:32:30,2017-10-24 22:32:30,63.30,94.40
   13418592130, 1009, *MAC address*, 2017-10-23 23:32:30,2017-10-24 23:32:30,95.51,50.17
   13418592131,1010, *MAC address*, 2017-10-24 00:32:30,2017-10-25 00:32:30,39.62,99.13

2. Use WinSCP to import the CSV file to the directory of the node where the client is installed, for example, **/opt**.
3. Log in to FusionInsight Manager and choose **System**. In the navigation pane on the left, choose **Permission** > **User**, click **Create** to create human-machine user **sparkuser**, and add the user to user groups hadoop (primary group) and hive.
4. Run the following commands to go to the client installation directory, load environment variables, and authenticate the user.

   cd /*Client installation directory*

   source ./bigdata_env

   source ./Spark2x/component_env

   NOTE

   In MRS 3.3.0-LTS and later versions, the Spark2x component is renamed Spark, and the role names of this component are also changed. For example, JobHistory2x is changed to JobHistory. Refer to the descriptions and operations related to the component name and role names in the document based on your MRS version.

   kinit sparkuser
5. Run the following command to upload the CSV file to the **/data** directory of the HDFS.

   hdfs dfs -put /opt/test.csv /data/

Connecting to CarbonData

- Use Spark SQL or Spark shell to connect to Spark and run Spark SQL commands.
- Run the following commands to start the JDBCServer and use a JDBC client (for example, Spark Beeline) to connect to the JDBCServer.

  cd ./Spark2x/spark/bin

  ./spark-beeline

Creating a CarbonData Table

After connecting Spark Beeline with the JDBCServer, create a CarbonData table to load data and perform query operations. Run the following commands to create a simple table:

create table x1 (imei string, deviceInformationId int, mac string, productdate timestamp, updatetime timestamp, gamePointId double, contractNumber

double) STORED AS carbondata TBLPROPERTIES ('SORT_COLUMNS'='imei,mac');

The command output is as follows:

```
+---------+
| Result  |
+---------+
+---------+
No rows selected (1.093 seconds)
```

Loading Data to a CarbonData Table

After you have created a CarbonData table, you can load the data from CSV to the created table.

Run the following command with required parameters to load data from CSV. The column names of the CarbonData table must match the column names of the CSV file.

LOAD DATA inpath 'hdfs://hacluster/data/*test.csv***' into table** ***x1*** **options('DELIMITER'=',', 'QUOTECHAR'='"','FILEHEADER'='imei, deviceinformationid,mac, productdate,updatetime, gamepointid,contractnumber');**

test.csv is the CSV file prepared in **Preparing a CSV File** and **x1** is the table name.

The CSV example file is as follows:

```
13418592122,1001, MAC address, 2017-10-23 15:32:30,2017-10-24 15:32:30,62.50,74.56
13418592123 1002,  MAC address, 2017-10-23 16:32:30,2017-10-24 16:32:30,17.80,76.28
13418592124,1003, MAC address, 2017-10-23 17:32:30,2017-10-24 17:32:30,20.40,92.94
13418592125 1004,  MAC address, 2017-10-23 18:32:30,2017-10-24 18:32:30,73.84,8.58
13418592126,1005, MAC address, 2017-10-23 19:32:30,2017-10-24 19:32:30,80.50,88.02
13418592127 1006,  MAC address, 2017-10-23 20:32:30,2017-10-24 20:32:30,65.77,71.24
13418592128,1007, MAC address, 2017-10-23 21:32:30,2017-10-24 21:32:30,75.21,76.04
13418592129,1008, MAC address, 2017-10-23 22:32:30,2017-10-24 22:32:30,63.30,94.40
13418592130, 1009, MAC address, 2017-10-23 23:32:30,2017-10-24 23:32:30,95.51,50.17
13418592131,1010, MAC address, 2017-10-24 00:32:30,2017-10-25 00:32:30,39.62,99.13
```

The command output is as follows:

```
+------------+
|Segment ID  |
+------------+
|0           |
+------------+
No rows selected (3.039 seconds)
```

Querying Data from a CarbonData Table

After a CarbonData table is created and the data is loaded, you can perform query operations as required. Some query operations are provided as examples.

- **Obtaining the number of records**

  Run the following command to obtain the number of records in the CarbonData table:

  select count(*) from x1;

- **Querying with the groupby condition**

  Run the following command to obtain the **deviceinformationid** records without repetition in the CarbonData table:

select deviceinformationid,count (distinct deviceinformationid) from x1 group by deviceinformationid;

- **Querying with Filter**

  Run the following command to obtain specific **deviceinformationid** records:

  select * from x1 where deviceinformationid='1010';

  NOTE

  If the query result has Chinese or other non-English characters, the columns in the query result may not be aligned. This is because characters of different languages occupy different widths.

Using CarbonData on Spark-shell

If you need to use CarbonData on a Spark-shell, you need to create a CarbonData table, load data to that table, and query data as follows:

```
spark.sql("CREATE TABLE x2(imei string, deviceInformationId int, mac string, productdate timestamp, updatetime timestamp, gamePointId double, contractNumber double) STORED AS carbondata")
spark.sql("LOAD DATA inpath 'hdfs://hacluster/data/x1_without_header.csv' into table x2 options('DELIMITER'=',', 'QUOTECHAR'='\"','FILEHEADER'='imei, deviceinformationid,mac, productdate,updatetime, gamepointid,contractnumber')")
spark.sql("SELECT * FROM x2").show()
```

1.4 CarbonData Data Analytics

1.4.1 Creating a CarbonData Table

Scenario

A CarbonData table must be created to load and query data. You can run the **Create Table** command to create a table. This command is used to create a table using custom columns.

Creating a Table with Self-Defined Columns

Users can create a table by specifying its columns and data types.

Sample command:

CREATE TABLE *IF NOT EXISTS productdb.productSalesTable (*

productNumber Int,

productName String,

storeCity String,

storeProvince String,

productCategory String,

productBatch String,

saleQuantity Int,

revenue Int)

STORED AS *carbondata*

TBLPROPERTIES (

'table_blocksize'='128');

The following table describes parameters of preceding commands.

Table 1-6 Parameter description

Parameter	Description
productSalesTable	Table name. The table is used to load data for analysis. The table name consists of letters, digits, and underscores (_).
productdb	Database name. The database maintains logical connections with tables stored in it to identify and manage the tables. The database name consists of letters, digits, and underscores (_).
productName storeCity storeProvince procuctCategory productBatch saleQuantity revenue	Columns in the table. The columns are service entities for data analysis. The column name (field name) consists of letters, digits, and underscores (_).
table_blocksize	Indicates the block size of data files used by the CarbonData table, in MB. The value ranges from **1** to **2048**. The default value is **1024**. If **table_blocksize** is too small, a large number of small files will be generated when data is loaded. This may affect the performance of HDFS. If **table_blocksize** is too large, during data query, the amount of block data that matches the index is large, and some blocks contain a large number of blocklets, affecting read concurrency and lowering query performance. You are advised to set the block size based on the data volume. For example, set the block size to 256 MB for GB-level data, 512 MB for TB-level data, and 1024 MB for PB-level data.

NOTE

- Measurement of all Integer data is processed and displayed using the **BigInt** data type.
- CarbonData parses data strictly. Any data that cannot be parsed is saved as **null** in the table. For example, if the user loads the **double** value (3.14) to the BigInt column, the data is saved as **null**.
- The Short and Long data types used in the **Create Table** command are shown as Smallint and BigInt in the **DESCRIBE** command, respectively.
- You can run the **DESCRIBE** command to view the table data size and table index size.

Operation Result

Run the command to create a table.

1.4.2 Deleting a CarbonData Table

Scenario

You can run the **DROP TABLE** command to delete a table. After a CarbonData table is deleted, its metadata and loaded data are deleted together.

Procedure

Run the following command to delete a CarbonData table:

Run the following command:

DROP TABLE *[IF EXISTS] [db_name.]table_name;*

Once this command is executed, the table is deleted from the system. In the command, **db_name** is an optional parameter. If **db_name** is not specified, the table named **table_name** in the current database is deleted.

Example:

DROP TABLE *productdb.productSalesTable;*

Run the preceding command to delete the **productSalesTable** table from the **productdb** database.

Operation Result

Deletes the table specified in the command from the system. After the table is deleted, you can run the **SHOW TABLES** command to check whether the table is successfully deleted. For details, see **SHOW TABLES**.

1.4.3 Modifying a CarbonData Table

SET and UNSET

When you run the SET command, all the attributes of the new SET overwrite the existing properties.

- SORT SCOPE

  An example of the SET SORT SCOPE command is as follows:

ALTER TABLE *tablename* **SET TBLPROPERTIES('SORT_SCOPE'**='*no_sort*')

After the UNSET SORT SCOPE command is executed, the default option NO_SORT is used.

Example for UNSET SORT SCOPE:

ALTER TABLE *tablename* **UNSET TBLPROPERTIES('SORT_SCOPE')**

- SORT COLUMNS

  Example for SET SORT COLUMNS:

  ALTER TABLE *tablename* **SET TBLPROPERTIES('SORT_COLUMNS'**='*column1*')

  This command applies new setting of SORT_COLUMNS to new imports. You can adjust SORT_COLUMNS to get optimal query result, but the old data will not be changed. Query performance of historical segments is not affected they are not queried based on the new SORT_COLUMNS.

  The UNSET command is not supported. Instead, you can set SORT_COLUMNS to an empty string.

  ALTER TABLE *tablename* **SET TBLPROPERTIES('SORT_COLUMNS'**='')

  NOTE

  - In later versions, custom merge will be enhanced to sort old segments again.
  - SORT_COLUMNS cannot be modified for streaming tables.
  - If an inverted index column is removed from SORT_COLUMNS, no inverted index will be created for the column. The old INVERTED_INDEX settings will not change.

1.4.4 Loading Data to a CarbonData Table

Scenario

After a CarbonData table is created, you can run the **LOAD DATA** command to load data to the table for query.

Once data loading is triggered, data is encoded in CarbonData format and files in multi-dimensional and column-based format are compressed and copied to the HDFS path of CarbonData files for quick analysis and queries.

The HDFS path can be configured in the **carbon.properties** file.

For details, see **Typical CarbonData Configuration Parameters**.

1.4.5 Deleting Segments of a CarbonData Table

Scenario

If you want to modify and reload the data because you have loaded wrong data into a table, or there are too many bad records, you can delete specific segments by segment ID or data loading time.

NOTE

The segment deletion operation only deletes segments that are not compacted. You can run the **CLEAN FILES** command to clear compacted segments.

Deleting a Segment by Segment ID

Each segment has a unique ID. This segment ID can be used to delete the segment.

Step 1 Obtain the segment ID.

Command:

SHOW SEGMENTS FOR Table *dbname.tablename LIMIT number_of_loads;*

Example:

SHOW SEGMENTS FOR TABLE *carbonTable;*

Run the preceding command to show all the segments of the table named **carbonTable**.

SHOW SEGMENTS FOR TABLE *carbonTable LIMIT 2;*

Run the preceding command to show segments specified by *number_of_loads*.

The command output is as follows:

```
+-----+----------+--------------------------+------------------+------------+------------+-------------+--------------+--
+
| ID  | Status |     Load Start Time      | Load Time Taken  | Partition  | Data Size  | Index Size  | File Format  |
+-----+----------+--------------------------+------------------+------------+------------+-------------+--------------+--
+
| 3   | Success  | 2020-09-28 22:53:26.336  | 3.726S           | {}         | 6.47KB     | 3.30KB      | columnar_v3  |
| 2   | Success  | 2020-09-28 22:53:01.702  | 6.688S           | {}         | 6.47KB     | 3.30KB      | columnar_v3  |
+-----+----------+--------------------------+------------------+------------+------------+-------------+--------------+--
+
```

NOTE

The output of the **SHOW SEGMENTS** command includes ID, Status, Load Start Time, Load Time Taken, Partition, Data Size, Index Size, and File Format. The latest loading information is displayed in the first line of the command output.

Step 2 Run the following command to delete the segment after you have found the Segment ID:

Command:

DELETE FROM TABLE tableName WHERE SEGMENT.ID IN (load_sequence_id1, load_sequence_id2, ...);

Example:

DELETE FROM TABLE carbonTable WHERE SEGMENT.ID IN (1,2,3);

For details, see **DELETE SEGMENT by ID**.

----End

Deleting a Segment by Data Loading Time

You can delete a segment based on the loading time.

Command:

DELETE FROM TABLE db_name.table_name WHERE SEGMENT.STARTTIME BEFORE date_value;

Example:

DELETE FROM TABLE carbonTable WHERE SEGMENT.STARTTIME BEFORE '2017-07-01 12:07:20';

The preceding command can be used to delete all segments before 2017-07-01 12:07:20.

For details, see **DELETE SEGMENT by DATE**.

Result

Data of corresponding segments is deleted and is unavailable for query. You can run the **SHOW SEGMENTS** command to display the segment status and check whether the segment has been deleted.

NOTE

- Segments are not physically deleted after the execution of the **DELETE SEGMENT** command. Therefore, if you run the **SHOW SEGMENTS** command to check the status of a deleted segment, it will be marked as **Marked for Delete**. If you run the **SELECT * FROM tablename** command, the deleted segment will be excluded.
- The deleted segment will be deleted physically only when the next data loading reaches the maximum query execution duration, which is configured by the **max.query.execution.time** parameter. The default value of the parameter is 60 minutes.
- If you want to forcibly delete a physical segment file, run the **CLEAN FILES** command.

  Example:

  CLEAN FILES FOR TABLE table1;

  This command will physically delete the segment file in the **Marked for delete** state.

  If this command is executed before the time specified by **max.query.execution.time** arrives, the query may fail. **max.query.execution.time** indicates the maximum time allowed for a query, which is set in the **carbon.properties** file.

1.4.6 Compacting CarbonData Table Segments

Scenario

Frequent data access results in a large number of fragmented CarbonData files in the storage directory. In each data loading, data is sorted and indexing is performed. This means that an index is generated for each load. With the increase of data loading times, the number of indexes also increases. As each index works only on one loading, the performance of index is reduced. CarbonData provides loading and compression functions. In a compression process, data in each segment is combined and sorted, and multiple segments are combined into one large segment.

Prerequisites

Multiple data loadings have been performed.

Operation Description

There are three types of compaction: Minor, Major, and Custom.

- Minor compaction:

  In minor compaction, you can specify the number of loads to be merged. If **carbon.enable.auto.load.merge** is set, minor compaction is triggered for every data load. If any segments are available to be merged, then compaction will run parallel with data load.

  There are two levels in minor compaction:

  - Level 1: Merging of the segments which are not yet compacted
  - Level 2: Merging of the compacted segments again to form a larger segment

- Major compaction:

  Multiple segments can be merged into one large segment. You can specify the compaction size so that all segments below the size will be merged. Major compaction is usually done during the off-peak time.

- Custom compaction:

  In Custom compaction, you can specify the IDs of multiple segments to merge them into a large segment. The IDs of all the specified segments must exist and be valid. Otherwise, the compaction fails. Custom compaction is usually done during the off-peak time.

For details, see **ALTER TABLE COMPACTION**.

Table 1-7 Compaction parameters

Parameter	Default Value	Application Type	Description
carbon.enable.auto.load.merge	false	Minor	Whether to enable compaction along with data loading. **true**: Compaction is automatically triggered when data is loaded. **false**: Compaction is not triggered when data is loaded.

Parameter	Default Value	Application Type	Description
carbon.compaction.level.threshold	4,3	Minor	This configuration is for minor compaction which decides how many segments to be merged. For example, if this parameter is set to **2,3**, minor compaction is triggered every two segments and segments form a single level 1 compacted segment. When the number of compacted level 1 segments reach 3, compaction is triggered again to merge them to form a single level 2 segment. The compaction policy depends on the actual data size and available resources. The value ranges from 0 to 100.
carbon.major.compaction.size	1024 MB	Major	The major compaction size can be configured using this parameter. Sum of the segments which is below this threshold will be merged. For example, if this parameter is set to 1024 MB, and there are five segments whose sizes are 300 MB, 400 MB, 500 MB, 200 MB, and 100 MB used for major compaction, only segments whose total size is less than this threshold are compacted. In this example, only the segments whose sizes are 300 MB, 400 MB, 200 MB, and 100 MB are compacted.
carbon.numberof.preserve.segments	0	Minor/ Major	If you want to preserve some number of segments from being compacted, then you can set this configuration. For example, if **carbon.numberof.preserve.segments** is set to **2**, the latest two segments will always be excluded from the compaction. By default, no segments are reserved.

Parameter	Default Value	Application Type	Description
carbon.allowed.compaction.days	0	Minor/ Major	This configuration is used to control on the number of recent segments that needs to be compacted. For example, if this parameter is set to **2**, the segments which are loaded in the time frame of past 2 days only will get merged. Segments which are loaded earlier than 2 days will not be merged. This configuration is disabled by default.
carbon.number.of.cores.while.compacting	2	Minor/ Major	Number of cores to be used while compacting data. The greater the number of cores, the better the compaction performance. If the CPU resources are sufficient, you can increase the value of this parameter.
carbon.merge.index.in.segment	true	SEGMENT_INDEX	If this parameter is set to **true**, all the Carbon index (.carbonindex) files in a segment will be merged into a single Index (.carbonindexmerge) file. This enhances the first query performance.

Reference

You are advised not to perform minor compaction on historical data. For details, see **How to Avoid Minor Compaction for Historical Data?**.

1.5 CarbonData Performance Tuning

1.5.1 Tuning Approach

Query Performance Tuning

There are various parameters that can be tuned to improve the query performance in CarbonData. Most of the parameters focus on increasing the parallelism in processing and optimizing system resource usage.

- Spark executor count: Executors are basic entities of parallelism in Spark. Raising the number of executors can increase the amount of parallelism in the cluster. For details about how to configure the number of executors, see the Spark documentation.

- Executor core: The number of concurrent tasks that an executor can run are controlled in each executor. Increasing the number of executor cores will add more concurrent processing tasks to improve performance.
- HDFS block size: CarbonData assigns query tasks by allocating different blocks to different executors for processing. HDFS block is the partition unit. CarbonData maintains a global block level index in Spark driver, which helps to reduce the quantity of blocks that need to be scanned for a query. Higher block size means higher I/O efficiency and lower global index efficiency. Reversely, lower block size means lower I/O efficiency, higher global index efficiency, and greater memory consumption.
- Number of scanner threads: Scanner threads control the number of parallel data blocks that are processed by each task. By increasing the number of scanner threads, you can increase the number of data blocks that are processed in parallel to improve performance. The **carbon.number.of.cores** parameter in the **carbon.properties** file is used to configure the number of scanner threads. For example, **carbon.number.of.cores = 4**.
- B-Tree caching: The cache memory can be optimized using the B-Tree least recently used (LRU) caching. In the driver, the B-Tree LRU caching configuration helps free up the cache by releasing table segments which are not accessed or not used. Similarly, in the executor, the B-Tree LRU caching configuration will help release table blocks that are not accessed or used. For details, see the description of **carbon.max.driver.lru.cache.size** and **carbon.max.executor.lru.cache.size** in **Table 1-18**.

CarbonData Query Process

When CarbonData receives a table query task, for example query for table A, the index data of table A will be loaded to the memory for the query process. When CarbonData receives a query task for table A again, the system does not need to load the index data of table A.

When a query is performed in CarbonData, the query task is divided into several scan tasks, namely, task splitting based on HDFS blocks. Scan tasks are executed by executors on the cluster. Tasks can run in parallel, partially parallel, or in sequence, depending on the number of executors and configured number of executor cores.

Some parts of a query task can be processed at the individual task level, such as **select** and **filter.** Some parts of a query task can be processed at the individual task level, such as **group-by**, **count**, and **distinct count**.

Some operations cannot be performed at the task level, such as **Having Clause** (filter after grouping) and **sort**. Operations which cannot be performed at the task level or can be only performed partially at the task level require data (partial results) transmission across executors on the cluster. The transmission operation is called shuffle.

The more the tasks are, the more data needs to be shuffled. This affects query performance.

The number of tasks is depending on the number of HDFS blocks and the number of blocks is depending on the size of each block. You are advised to configure proper HDFS block size to achieve a balance among increased parallelism, the amount of data to be shuffled, and the size of aggregate tables.

Relationship Between Splits and Executors

If the number of splits is less than or equal to the executor count multiplied by the executor core count, the tasks are run in parallel. Otherwise, some tasks can start only after other tasks are complete. Therefore, ensure that the executor count multiplied by executor cores is greater than or equal to the number of splits. In addition, make sure that there are sufficient splits so that a query task can be divided into sufficient subtasks to ensure concurrency.

Configuring Scanner Threads

The scanner threads property decides the number of data blocks to be processed. If there are too many data blocks, a large number of small data blocks will be generated, affecting performance. If there are few data blocks, the parallelism is poor and the performance is affected. When determining the number of scanner threads, you need to consider the average data size within a partition and select a value that makes the data block not small. Based on experience, you are advised to divide a single block size (unit: MB) by 250 and use the result as the number of scanner threads.

The number of actual available vCPUs is an important factor to consider when you want to increase the parallelism. The number of vCPUs that conduct parallel computation must not exceed 75% to 80% of actual vCPUs.

The number of vCPUs is approximately equal to:

Number of parallel tasks x Number of scanner threads. Number of parallel tasks is the smaller value of number of splits or executor count x executor cores.

Data Loading Performance Tuning

Tuning of data loading performance is different from that of query performance. Similar to query performance, data loading performance depends on the amount of parallelism that can be achieved. In case of data loading, the number of worker threads decides the unit of parallelism. Therefore, more executors mean more executor cores and better data loading performance.

To achieve better performance, you can configure the following parameters in HDFS.

Table 1-8 HDFS configuration

Parameter	Recommended Value
dfs.datanode.drop.cache.behind.reads	false
dfs.datanode.drop.cache.behind.writes	false
dfs.datanode.sync.behind.writes	true

Compression Tuning

CarbonData uses a few lightweight compression and heavyweight compression algorithms to compress data. Although these algorithms can process any type of

data, the compression performance is better if the data is ordered with similar values being together.

During data loading, data is sorted based on the order of columns in the table to achieve good compression performance.

Since CarbonData sorts data in the order of columns defined in the table, the order of columns plays an important role in the effectiveness of compression. If the low cardinality dimension is on the left, the range of data partitions after sorting is small and the compression efficiency is high. If a high cardinality dimension is on the left, a range of data partitions obtained after sorting is relatively large, and compression efficiency is relatively low.

Memory Tuning

CarbonData provides a mechanism for memory tuning where data loading depends on the columns needed in the query. Whenever a query command is received, columns required by the query are fetched and data is loaded for those columns in memory. During this operation, if the memory threshold is reached, the least used loaded files are deleted to release memory space for columns required by the query.

1.5.2 Typical Performance Tuning Parameters

Scenario

This section describes the configurations that can improve CarbonData performance.

Procedure

Table 1-9 and **Table 1-10** describe the configurations about query of CarbonData.

Table 1-9 Number of tasks started for the shuffle process

Parameter	spark.sql.shuffle.partitions
Configuration File	spark-defaults.conf
Function	Data query
Scenario Description	Number of tasks started for the shuffle process in Spark
Tuning	You are advised to set this parameter to one to two times as much as the executor cores. In an aggregation scenario, reducing the number from 200 to 32 can reduce the query time by two folds.

Table 1-10 Number of executors and vCPUs, and memory size used for CarbonData data query

Parameter	spark.executor.cores spark.executor.instances spark.executor.memory
Configuration File	spark-defaults.conf
Function	Data query
Scenario Description	Number of executors and vCPUs, and memory size used for CarbonData data query
Tuning	In the bank scenario, configuring 4 vCPUs and 15 GB memory for each executor will achieve good performance. The two values do not mean the more the better. Configure the two values properly in case of limited resources. If each node has 32 vCPUs and 64 GB memory in the bank scenario, the memory is not sufficient. If each executor has 4 vCPUs and 12 GB memory, Garbage Collection may occur during query, time spent on query from increases from 3s to more than 15s. In this case, you need to increase the memory or reduce the number of vCPUs.

Table 1-11, **Table 1-12**, and **Table 1-13** describe the configurations for CarbonData data loading.

Table 1-11 Number of vCPUs used for data loading

Parameter	carbon.number.of.cores.while.loading
Configuration File	carbon.properties
Function	Data loading
Scenario Description	Number of vCPUs used for data processing during data loading in CarbonData
Tuning	If there are sufficient CPUs, you can increase the number of vCPUs to improve performance. For example, if the value of this parameter is changed from 2 to 4, the CSV reading performance can be doubled.

Table 1-12 Whether to use Yarn local directories for multi-disk data loading

Parameter	carbon.use.local.dir

Configuration File	carbon.properties
Function	Data loading
Scenario Description	Whether to use Yarn local directories for multi-disk data loading
Tuning	If this parameter is set to **true**, CarbonData uses local Yarn directories for multi-table load disk load balance, improving data loading performance.

Table 1-13 Whether to use multiple directories during loading

Parameter	carbon.use.multiple.temp.dir
Configuration File	carbon.properties
Function	Data loading
Scenario Description	Whether to use multiple temporary directories to store temporary sort files
Tuning	If this parameter is set to **true**, multiple temporary directories are used to store temporary sort files during data loading. This configuration improves data loading performance and prevents single points of failure (SPOFs) on disks.

Table 1-14 describes the configurations for CarbonData data loading and query.

Table 1-14 Number of vCPUs used for data loading and query

Parameter	carbon.compaction.level.threshold
Configuration File	carbon.properties
Function	Data loading and query
Scenario Description	For minor compaction, specifies the number of segments to be merged in stage 1 and number of compacted segments to be merged in stage 2.

Tuning	Each CarbonData load will create one segment, if every load is small in size, it will generate many small files over a period of time impacting the query performance. Configuring this parameter will merge the small segments to one big segment which will sort the data and improve the performance. The compaction policy depends on the actual data size and available resources. For example, a bank loads data once a day and at night when no query is performed. If resources are sufficient, the compaction policy can be 6 or 5.

Table 1-15 Whether to enable data pre-loading when the index cache server is used

Parameter	carbon.indexserver.enable.prepriming
Configuration File	carbon.properties
Function	Data loading
Scenario Description	Enabling data pre-loading during the use of the index cache server can improve the performance of the first query.
Tuning	You can set this parameter to **true** to enable the pre-loading function. The default value is **false**.

1.5.3 Creating a CarbonData Table with High Query Performance

Scenario

This section provides suggestions based on more than 50 test cases to help you create CarbonData tables with higher query performance.

Table 1-16 Columns in the CarbonData table

Column name	Data type	Cardinality	Attribution
msname	String	30 million	dimension
BEGIN_TIME	bigint	10,000	dimension
host	String	1 million	dimension
dime_1	String	1,000	dimension
dime_2	String	500	dimension
dime_3	String	800	dimension

Column name	Data type	Cardinality	Attribution
counter_1	numeric(20,0)	NA	measure
...	...	NA	measure
counter_100	numeric(20,0)	NA	measure

Procedure

- If the to-be-created table contains a column that is frequently used for filtering, for example, this column is used in more than 80% of filtering scenarios,

  implement optimization as follows:

  Place this column in the first column of **sort_columns**.

  For example, if **msname** is used most frequently as a filter criterion in a query, it is placed in the first column. Run the following command to create a table. The query performance is good if **msname** is used as the filter criterion.

  ```
  create table carbondata_table(
     msname String,
     ...
     )STORED AS carbondata TBLPROPERTIES ('SORT_COLUMS'='msname');
  ```

- If the to-be-created table has multiple columns which are frequently used to filter the results,

  implement optimization as follows:

  Create an index for the columns.

  For example, if **msname**, **host**, and **dime_1** are frequently used columns, the **sort_columns** column sequence is "dime_1-> host-> msname..." based on cardinality. Run the following command to create a table. The following command can improve the filtering performance of **dime_1**, **host**, and **msname**.

  ```
  create table carbondata_table(
     dime_1 String,
     host String,
     msname String,
     dime_2 String,
     dime_3 String,
     ...
     )STORED AS carbondata
  TBLPROPERTIES ('SORT_COLUMS'='dime_1,host,msname');
  ```

- If the frequency of each column used for filtering is similar,

  implement optimization as follows:

  sort_columns is sorted in ascending order of cardinality.

  Run the following command to create a table:

  ```
  create table carbondata_table(
     Dime_1 String,
     BEGIN_TIME bigint,
     HOST String,
     msname String,
     ...
     )STORED AS carbondata
  TBLPROPERTIES ('SORT_COLUMS'='dime_2,dime_3,dime_1, BEGIN_TIME,host,msname');
  ```


- Create tables in ascending order of cardinalities. Then create secondary indexes for columns with more cardinalities. The statement for creating an index is as follows:

```
create index carbondata_table_index_msidn on tablecarbondata_table (
msname String) as 'carbondata' PROPERTIES ('table_blocksize'='128');
create index carbondata_table_index_host on tablecarbondata_table (
host String) as 'carbondata' PROPERTIES ('table_blocksize'='128');
```

- For columns of measure type, not requiring high accuracy, the numeric (20,0) data type is not required. You are advised to use the double data type to replace the numeric (20,0) data type to enhance query performance.

  The result of performance analysis of test-case shows reduction in query execution time from 15 to 3 seconds, thereby improving performance by nearly 5 times. The command for creating a table is as follows:

```
create table carbondata_table(
    Dime_1 String,
    BEGIN_TIME bigint,
    HOST String,
    msname String,
    counter_1 double,
    counter_2 double,
    ...
    counter_100 double,
    )STORED AS carbondata
;
```

- If values (**start_time** for example) of a column are incremental:

  For example, if data is loaded to CarbonData every day, **start_time** is incremental for each load. In this case, it is recommended that the **start_time** column be put at the end of **sort_columns**, because incremental values are efficient in using min/max index. The command for creating a table is as follows:

```
create table carbondata_table(
    Dime_1 String,
    HOST String,
    msname String,
    counter_1 double,
    counter_2 double,
    BEGIN_TIME bigint,
    ...
    counter_100 double,
    )STORED AS carbondata
    TBLPROPERTIES ( 'SORT_COLUMS'='dime_2,dime_3,dime_1..BEGIN_TIME');
```

1.6 Typical CarbonData Configuration Parameters

This section provides the details of how to configure all common CarbonData parameters.

Parameters in the carbon.properties File

Configure CarbonData parameters on the server or client based on the actual application scenario.

- Server: Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Spark**. Click **Configurations** then **All Configurations**, click **JDBCServer(Role)**, and select **Customization**. Then, add CarbonData parameters in **spark.carbon.customized.configs**.

- Client: Log in to the client node and configure related parameters in the *{Client installation directory}*/**Spark/spark/conf/carbon.properties** file.

Table 1-17 System configurations in **carbon.properties**

Parameter	Default Value	Description
carbon.ddl.base.hdfs.url	hdfs://hacluster/opt/data	HDFS relative path from the HDFS base path, which is configured in **fs.defaultFS**. The path configured in **carbon.ddl.base.hdfs.url** will be appended to the HDFS path configured in **fs.defaultFS**. If this path is configured, you do not need to pass the complete path while dataload. For example, if the absolute path of the CSV file is **hdfs://10.18.101.155:54310/data/cnbc/2016/xyz.csv**, the path **hdfs://10.18.101.155:54310** will come from property **fs.defaultFS** and you can configure **/data/cnbc/** as **carbon.ddl.base.hdfs.url**. During data loading, you can specify the CSV path as **/2016/xyz.csv**.
carbon.badRecords.location	-	Storage path of bad records. This path is an HDFS path. The default value is **Null**. If bad records logging or bad records operation redirection is enabled, the path must be configured by the user.
carbon.bad.records.action	fail	The following are four types of actions for bad records: **FORCE**: Data is automatically corrected by storing the bad records as NULL. **REDIRECT**: Bad records are written to the CSV file in **carbon.badRecords.location** instead of being loaded. **IGNORE**: Bad records are neither loaded nor written to the CSV file. **FAIL**: Data loading fails if any bad records are found.

Parameter	Default Value	Description
carbon.update.sync.folder	/tmp/carbondata	Specifies the **modifiedTime.mdt** file path. You can set it to an existing path or a new path. **NOTE** If you set this parameter to an existing path, ensure that all users can access the path and the path has the 777 permission.
carbon.enable.badrecord.action.redirect	false	Specifies whether to enable the REDIRECT mode to handle bad records during data loading. When it is enabled, bad records in source files will be recorded in a CSV file generated in a specified storage location each time data is loaded. CSV injection may occur when such CSV files are opened in Windows.
carbon.enable.partitiondata.trash	false	After this function is enabled, the **ALTER DROP PARTITION** operation moves the deleted partition data to the carbon trash. **NOTE** This function is supported only in MRS 3.2.0 or later.
carbon.enable.show.mv.for.showtables	false	If this parameter is set to **true**, materialized views are filtered out when the **show tables** command is executed. If this parameter is set to **true** when there are a large number of tables, the execution of the **show tables** command will take a long time. **NOTE** This function is supported only in MRS 3.2.0 or later.
carbon.enable.droptable.remove.staleentry	true	If this parameter is set to **true**, obsolete records of the table will be deleted from the cache when the **drop table** command is executed. If this parameter is set to **true** when there are a large number of databases, the execution of the **drop table** command will take a long time. **NOTE** This function is supported only in MRS 3.2.0 or later.

Parameter	Default Value	Description
carbon.enable.multi.version.table.status	false	Whether to enable versioning management of tablestatus files. If this parameter is set to **true**, a tablestatus file is generated each time a load, insert, or IUD operation is performed. **NOTE** If JDBCServer and the client are both used to load data to a table, ensure that this feature is enabled or disabled for both of them at the same time.
carbon.tablestatus.multi.version.file.coun	3	This parameter is available only when **carbon.enable.multi.version.table.status** is set to **true**. This parameter indicates the default number of the latest tablestatus files to be retained. Tablestatus files that exceed the value of this parameter will be deleted.

Table 1-18 Performance configurations in **carbon.properties**

Parameter	Default Value	Description
Data Loading Configuration		
carbon.sort.file.write.buffer.size	16384	CarbonData sorts data and writes it to a temporary file to limit memory usage. This parameter controls the size of the buffer used for reading and writing temporary files. The unit is bytes. The value ranges from 10240 to 10485760.
carbon.graph.rowset.size	100,000	Rowset size exchanged in data loading graph steps. The value ranges from 500 to 1,000,000.
carbon.number.of.cores.while.loading	6	Number of cores used during data loading. The greater the number of cores, the better the compaction performance. If the CPU resources are sufficient, you can increase the value of this parameter.
carbon.sort.size	500000	Number of records to be sorted

Parameter	Default Value	Description
carbon.enableXXHash	true	Hashmap algorithm used for hashkey calculation
carbon.number.of.cores.block.sort	7	Number of cores used for sorting blocks during data loading
carbon.max.driver.lru.cache.size	-1	Maximum size of LRU caching for data loading at the driver side. The unit is MB. The default value is **-1**, indicating that there is no memory limit for the caching. Only integer values greater than 0 are accepted.
carbon.max.executor.lru.cache.size	-1	Maximum size of LRU caching for data loading at the executor side. The unit is MB. The default value is **-1**, indicating that there is no memory limit for the caching. Only integer values greater than 0 are accepted. If this parameter is not configured, the value of **carbon.max.driver.lru.cache.size** is used.
carbon.merge.sort.prefetch	true	Whether to enable prefetch of data during merge sort while reading data from sorted temp files in the process of data loading
carbon.update.persist.enable	true	Configuration to enable the dataset of RDD/dataframe to persist data. Enabling this will reduce the execution time of UPDATE operation.
enable.unsafe.sort	true	Whether to use unsafe sort during data loading. Unsafe sort reduces the garbage collection during data load operation, resulting in better performance. The default value is **true**, indicating that unsafe sort is enabled.
enable.offheap.sort	true	Whether to use off-heap memory for sorting of data during data loading
offheap.sort.chunk.size.inmb	64	Size of data chunks to be sorted, in MB. The value ranges from 1 to 1024.

Parameter	Default Value	Description
carbon.unsafe.working.memory.in.mb	512	Size of the unsafe working memory. This will be used for sorting data and storing column pages. The unit is MB. Memory required for data loading: carbon.number.of.cores.while.loading [default value is 6] x Number of tables to load in parallel x offheap.sort.chunk.size.inmb [default value is 64 MB] + carbon.blockletgroup.size.in.mb [default value is 64 MB] + Current compaction ratio [64 MB/3.5] = Around 900 MB per table Memory required for data query: (SPARK_EXECUTOR_INSTANCES. [default value is 2] x (carbon.blockletgroup.size.in.mb [default value: 64 MB] + carbon.blockletgroup.size.in.mb [default value = 64 MB x 3.5) x Number of cores per executor [default value: 1]) = ~ 600 MB
carbon.sort.inmemory.storage.size.in.mb	512	Size of the intermediate sort data to be kept in the memory. Once the specified value is reached, the system writes data to the disk. The unit is MB.

Parameter	Default Value	Description
sort.inmemory.size.inmb	1024	Size of the intermediate sort data to be kept in the memory. Once the specified value is reached, the system writes data to the disk. The unit is MB. If **carbon.unsafe.working.memory.in.mb** and **carbon.sort.inmemory.storage.size.in.mb** are configured, you do not need to set this parameter. If this parameter has been configured, 20% of the memory is used for working memory **carbon.unsafe.working.memory.in.mb**, and 80% is used for sort storage memory **carbon.sort.inmemory.storage.size.in.mb**. **NOTE** The value of **spark.yarn.executor.memoryOverhead** configured for Spark must be greater than the value of **sort.inmemory.size.inmb** configured for CarbonData. Otherwise, Yarn might stop the executor if off-heap access exceeds the configured executor memory.
carbon.blockletgroup.size.in.mb	64	The data is read as a group of blocklets which are called blocklet groups. This parameter specifies the size of each blocklet group. Higher value results in better sequential I/O access. The minimum value is 16 MB. Any value less than 16 MB will be reset to the default value (64 MB). The unit is MB.
enable.inmemory.merge.sort	false	Whether to enable **inmemorymerge sort**.
use.offheap.in.query.processing	true	Whether to enable **offheap** in query processing.

Parameter	Default Value	Description
carbon.load.sort.scope	local_sort	Sort scope for the load operation. There are two types of sort: **batch_sort** and **local_sort**. If **batch_sort** is selected, the loading performance is improved but the query performance is reduced. **NOTE** **local_sort** conflicts with DDL operations on partitioned tables and they cannot be used at the same time. In addition, **local_sort** does not significantly improve the performance of partitioned tables. You are advised not to enable this feature on partitioned tables.
carbon.batch.sort.size.inmb	-	Size of data to be considered for batch sorting during data loading. The recommended value is less than 45% of the total sort data. The unit is MB. **NOTE** If this parameter is not set, its value is about 45% of the value of **sort.inmemory.size.inmb** by default.
enable.unsafe.columnpage	true	Whether to keep page data in heap memory during data loading or query to prevent garbage collection bottleneck.
carbon.use.local.dir	false	Whether to use Yarn local directories for multi-disk data loading. If this parameter is set to **true**, Yarn local directories are used to load multi-disk data to improve data loading performance.
carbon.use.multiple.temp.dir	false	Whether to use multiple temporary directories for storing temporary files to improve data loading performance.
carbon.load.datamaps.parallel.db_name.table_name	N/A	The value can be **true** or **false**. You can set the database name and table name to improve the first query performance of the table.
Compaction Configuration		
carbon.number.of.cores.while.compacting	2	Number of cores to be used while compacting data. The greater the number of cores, the better the compaction performance. If the CPU resources are sufficient, you can increase the value of this parameter.

Parameter	Default Value	Description
carbon.compaction.level.threshold	4,3	This configuration is for minor compaction which decides how many segments to be merged. For example, if this parameter is set to **2,3**, minor compaction is triggered every two segments. **3** is the number of level 1 compacted segments which is further compacted to new segment. The value ranges from 0 to 100.
carbon.major.compaction.size	1024	Major compaction size. Sum of the segments which is below this threshold will be merged. The unit is MB.
carbon.horizontal.compaction.enable	true	Whether to enable/disable horizontal compaction. After every DELETE and UPDATE statement, horizontal compaction may occur in case the incremental (DELETE/ UPDATE) files becomes more than specified threshold. By default, this parameter is set to **true**. You can set this parameter to **false** to disable horizontal compaction.
carbon.horizontal.update.compaction.threshold	1	Threshold limit on number of UPDATE delta files within a segment. In case the number of delta files goes beyond the threshold, the UPDATE delta files within the segment becomes eligible for horizontal compaction and are compacted into single UPDATE delta file. By default, this parameter is set to **1**. The value ranges from **1** to **10000**.
carbon.horizontal.delete.compaction.threshold	1	Threshold limit on number of DELETE incremental files within a block of a segment. In case the number of incremental files goes beyond the threshold, the DELETE incremental files for the particular block of the segment becomes eligible for horizontal compaction and are compacted into single DELETE incremental file. By default, this parameter is set to **1**. The value ranges from **1** to **10000**.
Query Configuration		

Parameter	Default Value	Description
carbon.number.of.cores	4	Number of cores to be used during query
carbon.limit.block.distribution.enable	false	Whether to enable the CarbonData distribution for limit query. The default value is **false**, indicating that block distribution is disabled for query statements that contain the keyword limit. For details about how to optimize this parameter, see **Typical Performance Tuning Parameters**.
carbon.custom.block.distribution	false	Whether to enable Spark or CarbonData block distribution. By default, the value is **false**, indicating that Spark block distribution is enabled. To enable CarbonData block distribution, change the value to **true**.
carbon.infilter.subquery.pushdown.enable	false	If this is set to **true** and a Select query is triggered in the filter with subquery, the subquery is executed and the output is broadcast as IN filter to the left table. Otherwise, SortMergeSemiJoin is executed. You are advised to set this to **true** when IN filter subquery does not return too many records. For example, when the IN clause subquery returns 10,000 or fewer records, enabling this parameter will display query results faster. Example: ***select * from flow_carbon_256b where cus_no in (select cus_no from flow_carbon_256b where dt>='20260101' and dt<='20260701' and txn_bk='tk_1' and txn_br='tr_1') limit 1000;***
carbon.scheduler.minRegisteredResourcesRatio	0.8	Minimum resource (executor) ratio needed for starting the block distribution. The default value is **0.8**, indicating that 80% of the requested resources are allocated for starting block distribution.
carbon.dynamicAllocation.schedulerTimeout	5	Maximum time that the scheduler waits for executors to be active. The default value is **5** seconds, and the maximum value is **15** seconds.

Parameter	Default Value	Description
enable.unsafe.in.query.processing	true	Whether to use unsafe sort during query. Unsafe sort reduces the garbage collection during query, resulting in better performance. The default value is **true**, indicating that unsafe sort is enabled.
carbon.enable.vector.reader	true	Whether to enable vector processing for result collection to improve query performance
carbon.query.show.datamaps	true	**SHOW TABLES** lists all tables, including the primary table and datamaps. To filter out the datamaps, set this parameter to **false**.
Secondary Index Configuration		
carbon.secondary.index.creation.threads	1	Number of threads to concurrently process segments during secondary index creation. This property helps fine-tuning the system when there are a lot of segments in a table. The value ranges from 1 to 50.
carbon.si.lookup.partialstring	true	• When the parameter value is **true**, it includes indexes started with, ended with, and contained. • When the parameter value is **false**, it includes only secondary indexes started with.
carbon.si.segment.merge	true	Enabling this property merges **.carbondata** files inside the secondary index segment. The merging will happen after the load operation. That is, at the end of the secondary index table load, small files are checked and merged. **NOTE** Table Block Size is used as the size threshold for merging small files.

Table 1-19 Other configurations in **carbon.properties**

Parameter	Default Value	Description
Data Loading Configuration		

Parameter	Default Value	Description
carbon.lock.type	HDFSLOCK	Type of lock to be acquired during concurrent operations on a table. There are following types of lock implementation: • **LOCALLOCK**: Lock is created on local file system as a file. This lock is useful when only one Spark driver (or JDBCServer) runs on a machine. • **HDFSLOCK**: Lock is created on HDFS file system as a file. This lock is useful when multiple Spark applications are running and no ZooKeeper is running on a cluster.
carbon.sort.intermediate.files.limit	20	Minimum number of intermediate files. After intermediate files are generated, sort and merge the files. For details about how to optimize this parameter, see **Typical Performance Tuning Parameters**.
carbon.csv.read.buffersize.byte	1048576	Size of CSV reading buffer
carbon.merge.sort.reader.thread	3	Maximum number of threads used for reading intermediate files for final merging.
carbon.concurrent.lock.retries	100	Maximum number of retries used to obtain the concurrent operation lock. This parameter is used for concurrent loading.
carbon.concurrent.lock.retry.timeout.sec	1	Interval between the retries to obtain the lock for concurrent operations.
carbon.lock.retries	3	Maximum number of retries to obtain the lock for any operations other than import.
carbon.lock.retry.timeout.sec	5	Interval between the retries to obtain the lock for any operation other than import.

Parameter	Default Value	Description
carbon.tempstore.location	/opt/Carbon/TempStoreLoc	Temporary storage location. By default, the **System.getProperty("java.io.tmpdir")** method is used to obtain the value. For details about how to optimize this parameter, see the description of **carbon.use.local.dir** in **Typical Performance Tuning Parameters**.
carbon.load.log.counter	500000	Data loading records count in logs
SERIALIZATION_NULL_FORMAT	\N	Value to be replaced with NULL
carbon.skip.empty.line	false	Setting this property will ignore the empty lines in the CSV file during data loading.
carbon.load.datamaps.parallel	false	Whether to enable parallel datamap loading for all tables in all sessions. This property will improve the time to load datamaps into memory by distributing the job among executors, thus improving query performance.
Merging Configuration		
carbon.numberof.preserve.segments	0	If you want to preserve some number of segments from being compacted, then you can set this configuration. For example, if **carbon.numberof.preserve.segments** is set to **2**, the latest two segments will always be excluded from the compaction. No segments will be preserved by default.
carbon.allowed.compaction.days	0	This configuration is used to control on the number of recent segments that needs to be merged. For example, if this parameter is set to **2**, the segments which are loaded in the time frame of past 2 days only will get merged. Segments which are loaded earlier than 2 days will not be merged. This configuration is disabled by default.

Parameter	Default Value	Description
carbon.enable.auto.load.merge	false	Whether to enable compaction along with data loading.
carbon.merge.index.in.segment	true	This configuration merges all the CarbonIndex files (**.carbonindex**) in a segment into a single MergeIndex file (**.carbonindexmerge**) upon data loading completion. This significantly reduces the delay in serving the first query.
carbon.enable.compact.autoclean	false	If this parameter is set to **true**, the **clean files** command is invoked to delete obsolete files after segments are compacted. **NOTE** This function is supported only in MRS 3.2.0 or later.
Query Configuration		
max.query.execution.time	60	Maximum time allowed for one query to be executed. The unit is minute.
carbon.enableMinMax	true	MinMax is used to improve query performance. You can set this to **false** to disable this function.
carbon.lease.recovery.retry.count	5	Maximum number of attempts that need to be made for recovering a lease on a file. Minimum value: **1** Maximum value: **50**
carbon.lease.recovery.retry.interval	1000 (ms)	Interval or pause time after a lease recovery attempt is made on a file. Minimum value: **1000** (ms) Maximum value: **10000** (ms)

Parameters in the spark-defaults.conf File

- Log in to the client node and configure the parameters listed in **Table 1-20** in the *{Client installation directory}*/**Spark/spark/conf/spark-defaults.conf** file.

Table 1-20 Spark configuration reference in **spark-defaults.conf**

Parameter	Default Value	Description
spark.driver.memory	4G	Memory to be used for the driver process. SparkContext has been initialized. **NOTE** In client mode, do not use SparkConf to set this parameter in the application because the driver JVM has been started. To configure this parameter, configure it in the **--driver-memory** command-line option or in the default property file.
spark.executor.memory	4 GB	Memory to be used for each executor process.
spark.sql.crossJoin.enabled	true	If the query contains a cross join, enable this property so that no error occurs. In this case, you can use a cross join instead of a join for better performance.

- Configure the following parameters in the **spark-defaults.conf** file on the Spark driver.
  - In spark-sql mode: Log in to the Spark client node and configure the parameters listed in **Table 1-21** in the *{Client installation directory}*/**Spark/spark/conf/spark-defaults.conf** file.

Table 1-21 Parameter description

Parameter	Value	Description
spark.driver.extraJavaOptions	-Dlog4j.configuration=file:/opt/client/Spark/spark/conf/log4j.properties -Djetty.version=x.y.z -Dzookeeper.server.principal=zookeeper/hadoop.*<System domain name>* -Djava.security.krb5.conf=/opt/client/KrbClient/kerberos/var/krb5kdc/krb5.conf -Djava.security.auth.login.config=/opt/client/Spark/spark/conf/jaas.conf -Dorg.xerial.snappy.tempdir=/opt/client/Spark/tmp -Dcarbon.properties.filepath=/opt/client/Spark/spark/conf/carbon.properties -Djava.io.tmpdir=/opt/client/Spark/tmp	The default value **/opt/client/Spark/spark** indicates **CLIENT_HOME** of the client and is added to the end of the value of **spark.driver.extraJavaOptions**. This parameter is used to specify the path of the **carbon.properties**file in Driver. **NOTE** Spaces next to equal marks (=) are not allowed.
spark.sql.session.state.builder	org.apache.spark.sql.hive.FIHiveACLSessionStateBuilder	Session state constructor.
spark.carbon.sqlastbuilder.classname	org.apache.spark.sql.hive.CarbonInternalSqlAst-Builder	AST constructor.

Parameter	Value	Description
spark.sql.catalog.class	org.apache.spark.sql.hive.HiveACLExternalCatalog	Hive External catalog to be used. This parameter is mandatory if Spark ACL is enabled.
spark.sql.hive.implementation	org.apache.spark.sql.hive.HiveACLClientImpl	How to call the Hive client. This parameter is mandatory if Spark ACL is enabled.
spark.sql.hiveClient.isolation.enabled	false	This parameter is mandatory if Spark ACL is enabled.

- In JDBCServer: Log in to the node where JDBCServer is installed and configure the parameters listed in **Table 1-22** in the **{BIGDATA_HOME}/FusionInsight_Spark_*/*_JDBCServer/etc/spark-defaults.conf** file.

Table 1-22 Parameter description

Parameter	Value	Description
spark.driver.extraJavaOptions	-Xloggc:${SPARK_LOG_DIR}/indexserver-omm-%p-gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:MaxDirectMemorySize=512M -XX:MaxMetaspaceSize=512M -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -XX:OnOutOfMemoryError='kill -9 %p' -Djetty.version=x.y.z -Dorg.xerial.snappy.tempdir=${BIGDATA_HOME}/tmp/spark/JDBCServer/snappy_tmp -Djava.io.tmpdir=${BIGDATA_HOME}/tmp/spark/JDBCServer/io_tmp -Dcarbon.properties.filepath=${SPARK_CONF_DIR}/carbon.properties -Djdk.tls.ephemeralDHKeySize=2	The default value **${SPARK_CONF_DIR}** depends on a specific cluster and is added to the end of the value of the **spark.driver.extraJavaOptions** parameter. This parameter is used to specify the path of the **carbon.properties** file in Driver. **NOTE** Spaces next to equal marks (=) are not allowed.

Parameter	Value	Description
	048 -Dspark.ssl.keyStore=${SPARK_CONF_DIR}/child.keystore #{java_stack_prefer}	
spark.sql.session.state.builder	org.apache.spark.sql.hive.FIHiveACLSessionStateBuilder	Session state constructor.
spark.carbon.sqlastbuilder.classname	org.apache.spark.sql.hive.CarbonInternalSqlAst-Builder	AST constructor.
spark.sql.catalog.class	org.apache.spark.sql.hive.HiveACLExternalCatalog	Hive External catalog to be used. This parameter is mandatory if Spark ACL is enabled.
spark.sql.hive.implementation	org.apache.spark.sql.hive.HiveACLClientImpl	How to call the Hive client. This parameter is mandatory if Spark ACL is enabled.
spark.sql.hiveClient.isolation.enabled	false	This parameter is mandatory if Spark ACL is enabled.

1.7 CarbonData Syntax Reference

1.7.1 CREATE TABLE

Function

This command is used to create a CarbonData table by specifying the list of fields along with the table properties.

Syntax

CREATE TABLE *[IF NOT EXISTS] [db_name.]table_name*

[(col_name data_type, ...)]

STORED AS *carbondata*

[TBLPROPERTIES (property_name=property_value, ...)];

Additional attributes of all tables are defined in **TBLPROPERTIES**.

Parameter Description

Table 1-23 CREATE TABLE parameters

Parameter	Description
db_name	Database name that contains letters, digits, and underscores (_).
col_name data_type	List with data types separated by commas (,). The column name contains letters, digits, and underscores (_). **NOTE** When creating a CarbonData table, do not use tupleId, PositionId, and PositionReference as column names because columns with these names are internally used by secondary index commands.
table_name	Table name of a database that contains letters, digits, and underscores (_).
STORED AS	The **carbondata** parameter defines and creates a CarbonData table.
TBLPROPERTIES	List of CarbonData table properties.

Precautions

Table attributes are used as follows:

- Block size

  The block size of a data file can be defined for a single table using **TBLPROPERTIES**. The larger one between the actual size of the data file and the defined block size is selected as the actual block size of the data file in HDFS. The unit is MB. The default value is 1024 MB. The value ranges from 1 MB to 2048 MB. If the value is beyond the range, the system reports an error.

  Once the block size reaches the configured value, the write program starts a new block of CarbonData data. Data is written in multiples of the page size (32,000 records). Therefore, the boundary is not strict at the byte level. If the new page crosses the boundary of the configured block, the page is written to the new block instead of the current block.

  TBLPROPERTIES('table_blocksize'='128')

NOTE

- If a small block size is configured in the CarbonData table while the size of the data file generated by the loaded data is large, the block size displayed in HDFS is different from the configured value. This is because when data is written to a local block file for the first time, even though the size of the to-be-written data is larger than the configured value of the block size, data will still be written into the block. Therefore, the actual value of block size in HDFS is the larger value between the size of the data to be written and the configured block size.
- If **block.num** is less than the parallelism, the blocks are split into new blocks so that new blocks.num is greater than parallelism and all cores can be used. This optimization is called block distribution.

- **SORT_SCOPE** specifies the sort scope during table creation. There are four types of sort scopes:
  - **GLOBAL_SORT**: It improves query performance, especially for point queries. *TBLPROPERTIES('SORT_SCOPE'='GLOBAL_SORT'*)
  - **LOCAL_SORT**: Data is sorted locally (task-level sorting).
  - **NO_SORT**: The default sorting mode is used. Data is loaded in unsorted manner, which greatly improves loading performance.
- SORT_COLUMNS

  This table property specifies the order of sort columns.

  TBLPROPERTIES('SORT_COLUMNS'='column1, column3')

  NOTE

  - If this attribute is not specified, no columns are sorted by default.
  - If this property is specified but with empty argument, then the table will be loaded without sort. For example, *('SORT_COLUMNS'='')*.
  - **SORT_COLUMNS** supports the string, date, timestamp, short, int, long, byte, and boolean data types.
- RANGE_COLUMN

  This property is used to specify a column to partition the input data by range. Only one column can be configured. During data import, you can use **global_sort_partitions** or **scale_factor** to avoid generating small files.

  TBLPROPERTIES('RANGE_COLUMN'='column1')
- LONG_STRING_COLUMNS

  The length of a common string cannot exceed 32,000 characters. To store a string of more than 32,000 characters, set **LONG_STRING_COLUMNS** to the target column.

  TBLPROPERTIES('LONG_STRING_COLUMNS'='column1, column3')

  NOTE

  LONG_STRING_COLUMNS can be set only for columns of the STRING, CHAR, or VARCHAR type.

Scenarios

Creating a Table by Specifying Columns

The **CREATE TABLE** command is the same as that of Hive DDL. The additional configurations of CarbonData are provided as table properties.

CREATE TABLE *[IF NOT EXISTS] [db_name.]table_name*

[(col_name data_type , ...)]

STORED AS *carbondata*

[TBLPROPERTIES (property_name=property_value, ...)];

Examples

CREATE TABLE *IF NOT EXISTS productdb.productSalesTable (*

productNumber Int,

productName String,

storeCity String,

storeProvince String,

productCategory String,

productBatch String,

saleQuantity Int,

revenue Int)

STORED AS carbondata

TBLPROPERTIES (

'table_blocksize'='128',

'SORT_COLUMNS'='productBatch, productName')

System Response

A table will be created and the success message will be logged in system logs.

1.7.2 CREATE TABLE As SELECT

Function

This command is used to create a CarbonData table by specifying the list of fields along with the table properties.

Syntax

CREATE TABLE *[IF NOT EXISTS] [db_name.]table_name* ***STORED AS carbondata*** *[TBLPROPERTIES (key1=val1, key2=val2, ...)] AS select_statement;*

Parameter Description

Table 1-24 CREATE TABLE parameters

Parameter	Description
db_name	Database name that contains letters, digits, and underscores (_).
table_name	Table name of a database that contains letters, digits, and underscores (_).
STORED AS	Used to store data in CarbonData format.
TBLPROPERTIES	List of CarbonData table properties. For details, see **Precautions**.

Precautions

N/A

Examples

CREATE TABLE ctas_select_parquet ***STORED AS*** carbondata as select * from parquet_ctas_test;

System Response

This example will create a Carbon table from any Parquet table and load all the records from the Parquet table.

1.7.3 DROP TABLE

Function

This command is used to delete an existing table.

Syntax

DROP TABLE *[IF EXISTS] [db_name.]table_name;*

Parameter Description

Table 1-25 DROP TABLE parameters

Parameter	Description
db_name	Database name. If this parameter is not specified, the current database is selected.
table_name	Name of the table to be deleted

Precautions

In this command, **IF EXISTS** and **db_name** are optional.

Example

DROP TABLE IF EXISTS productDatabase.productSalesTable;

System Response

The table will be deleted.

1.7.4 SHOW TABLES

Function

SHOW TABLES command is used to list all tables in the current or a specific database.

Syntax

SHOW TABLES *[IN db_name];*

Parameter Description

Table 1-26 SHOW TABLE parameters

Parameter	Description
IN db_name	Name of the database. This parameter is required only when tables of this specific database are to be listed.

Usage Guidelines

IN db_Name is optional.

Examples

SHOW TABLES IN ProductDatabase;

System Response

All tables are listed.

1.7.5 ALTER TABLE COMPACTION

Function

The **ALTER TABLE COMPACTION** command is used to merge a specified number of segments into a single segment. This improves the query performance of a table.

Syntax

ALTER TABLE*[db_name.]table_name COMPACT 'MINOR/MAJOR/SEGMENT_INDEX';*

ALTER TABLE*[db_name.]table_name COMPACT 'CUSTOM' WHERE SEGMENT.ID IN (id1, id2, ...);*

Parameter Description

Table 1-27 ALTER TABLE COMPACTION parameters

Parameter	Description
db_name	Database name. If this parameter is not specified, the current database is selected.
table_name	Table name.
MINOR	Minor compaction. For details, see **Compacting CarbonData Table Segments**.
MAJOR	Major compaction. For details, see **Compacting CarbonData Table Segments**.
SEGMENT_INDEX	This configuration enables you to merge all the CarbonData index files (**.carbonindex**) inside a segment to a single CarbonData index merge file (**.carbonindexmerge**). This enhances the first query performance. For more information, see **Table 1-7**.
CUSTOM	Custom compaction. For details, see **Combining Segments**.

Precautions

N/A

Examples

ALTER TABLE ProductDatabase COMPACT 'MINOR';

ALTER TABLE ProductDatabase COMPACT 'MAJOR';

ALTER TABLE ProductDatabase COMPACT 'SEGMENT_INDEX';

ALTER TABLE ProductDatabase COMPACT 'CUSTOM' WHERE SEGMENT.ID IN (0, 1);

System Response

ALTER TABLE COMPACTION does not show the response of the compaction because it is run in the background.

If you want to view the response of minor and major compactions, you can check the logs or run the ***SHOW SEGMENTS*** command.

Example:

```
+------+------------+--------------------------+-----------------+------------+------------+-------------+--------------
+--+
|  ID  |   Status   |     Load Start Time      | Load Time Taken  | Partition  | Data Size  | Index Size  | File
Format  |
+------+------------+--------------------------+-----------------+------------+------------+-------------+--------------
+--+
| 3    | Success    | 2020-09-28 22:53:26.336  | 3.726S           | {}         | 6.47KB     | 3.30KB      | columnar_v3  |
| 2    | Success    | 2020-09-28 22:53:01.702  | 6.688S           | {}         | 6.47KB     | 3.30KB      | columnar_v3  |
| 1    | Compacted  | 2020-09-28 22:51:15.242  | 5.82S            | {}         | 6.50KB     | 3.43KB      |
columnar_v3  |
| 0.1  | Success    | 2020-10-30 20:49:24.561  | 16.66S           | {}         | 12.87KB    | 6.91KB      | columnar_v3
|
| 0    | Compacted  | 2020-09-28 22:51:02.6    | 6.819S           | {}         | 6.50KB     | 3.43KB      | columnar_v3
|
+------+------------+--------------------------+-----------------+------------+------------+-------------+--------------
+--+
```

In the preceding information:

- **Compacted** indicates that data has been compacted.
- **0.1** indicates the compacting result of segment 0 and segment 1.

The compact operation does not incur any change to other operations.

Compacted segments, such as segment 0 and segment 1, become useless. To save space, before you perform other operations, run the ***CLEAN FILES*** command to delete compacted segments. For more information about the **CLEAN FILES** command, see **CLEAN FILES**.

1.7.6 TABLE RENAME

Function

This command is used to rename an existing table.

Syntax

ALTER TABLE *[db_name.]table_name* **RENAME TO** *new_table_name*;

Parameter Description

Table 1-28 RENAME parameters

Parameter	Description
db_name	Database name. If this parameter is not specified, the current database is selected.
table_name	Current name of the existing table
new_table_name	New name of the existing table

Precautions

- Parallel queries (using table names to obtain paths for reading CarbonData storage files) may fail during this operation.
- The secondary index table cannot be renamed.

Example

ALTER TABLE *carbon* **RENAME TO** *carbondata*;

ALTER TABLE *test_db.carbon* **RENAME TO** *test_db.carbondata*;

System Response

The new table name will be displayed in the CarbonData folder. You can run **SHOW TABLES** to view the new table name.

1.7.7 ADD COLUMNS

Function

This command is used to add a column to an existing table.

Syntax

ALTER TABLE *[db_name.]table_name* **ADD COLUMNS** *(col_name data_type,...)* **TBLPROPERTIES***(''COLUMNPROPERTIES.columnName.shared_column'='sharedFolder.sharedColumnName,...', 'DEFAULT.VALUE.COLUMN_NAME'='default_value')*;

Parameter Description

Table 1-29 ADD COLUMNS parameters

Parameter	Description
db_name	Database name. If this parameter is not specified, the current database is selected.
table_name	Table name.
col_name data_type	Name of a comma-separated column with a data type. It consists of letters, digits, and underscores (_). **NOTE** When creating a CarbonData table, do not name columns as tupleId, PositionId, and PositionReference because they will be used in UPDATE, DELETE, and secondary index commands.

Precautions

- Only **shared_column** and **default_value** are read. If any other property name is specified, no error will occur and the property will be ignored.

- If no default value is specified, the default value of the new column is considered null.
- If filter is applied to the column, new columns will not be added during sort. New columns may affect query performance.

Examples

- **ALTER TABLE** *carbon* **ADD COLUMNS** *(a1 INT, b1 STRING)*;
- **ALTER TABLE** *carbon* **ADD COLUMNS** *(a1 INT, b1 STRING)* **TBLPROPERTIES***('COLUMNPROPERTIES.b1.shared_column'='sharedFolder.b1')*;
- ALTER TABLE *carbon* **ADD COLUMNS** *(a1 INT, b1 STRING)* **TBLPROPERTIES***('DEFAULT.VALUE.a1'='10')*;

System Response

The newly added column can be displayed by running the **DESCRIBE** command.

1.7.8 DROP COLUMNS

Function

This command is used to delete one or more columns from a table.

Syntax

ALTER TABLE *[db_name.]table_name* **DROP COLUMNS** *(col_name, ...)*;

Parameter Description

Table 1-30 DROP COLUMNS parameters

Parameter	Description
db_name	Database name. If this parameter is not specified, the current database is selected.
table_name	Table name.
col_name	Name of a column in a table. Multiple columns are supported. It consists of letters, digits, and underscores (_).

Precautions

After a column is deleted, at least one key column must exist in the schema. Otherwise, an error message is displayed, and the column fails to be deleted.

Examples

Assume that the table contains four columns named a1, b1, c1, and d1.

- Delete a column:
  ALTER TABLE *carbon* **DROP COLUMNS** *(b1)*;
  ALTER TABLE *test_db.carbon* **DROP COLUMNS** *(b1)*;
- Delete multiple columns:
  ALTER TABLE *carbon* **DROP COLUMNS** *(b1,c1)*;
  ALTER TABLE *test_db.carbon* **DROP COLUMNS** *(b1,c1)*;

System Response

If you run the **DESCRIBE** command, the deleted columns will not be displayed.

1.7.9 CHANGE DATA TYPE

Function

This command is used to convert INT to BIGINT or increase decimal precision.

Syntax

ALTER TABLE *[db_name.]table_name* **CHANGE** *col_name col_name changed_column_type*;

Parameter Description

Table 1-31 CHANGE DATA TYPE parameters

Parameter	Description
db_name	Database name. If this parameter is not specified, the current database is selected.
table_name	Table name.
col_name	Name of a column in a table. A column name consists of letters, digits, and underscores (_).
changed_column_type	Target data type.

Precautions

- The decimal precision conversion is available only when no data is lost. The decimal precision is represented in the (precision, scale) format.
  - Unavailable: Decimal precision cannot be changed from (10,2) to (10,5) because only the scale increases, but the total number of digits remains unchanged.
  - Available: Decimal precision can be changed from (10,2) to (12,3). The total number of digits increases by 2 but the scale increases only by 1, which does not cause any data loss.
- The maximum precision allowed is (38,38).

Example

- Convert the data type of column **a1** from INT to BIGINT.
  ALTER TABLE *test_db.carbon* **CHANGE** *a1 a1 BIGINT*;
- Change the precision of column **a1** from 10 to 18.
  ALTER TABLE *test_db.carbon* **CHANGE** *a1 a1 DECIMAL(18,2)*;

Response

You can run the **DESCRIBE** command to check the changed data type of the modified column.

1.7.10 REFRESH TABLE

Function

This command is used to register Carbon table data with the Hive metabase.

Syntax

REFRESH TABLE *db_name.table_name*;

Parameter Description

Table 1-32 REFRESH TABLE parameters

Parameter	Description
db_name	Database name. If this parameter is not specified, the current database is selected.
table_name	Table name.

Precautions

- Before you run this command, copy the schema and data of the source table to the destination database.
- For the repository of an earlier version, the time zones of the source and destination clusters must be the same.
- The names of the destination database and the source database must be the same.
- If you are using aggregate tables, all of them should be copied to the destination database.
- If the source cluster uses Hive metabase to store the table structure, the refresh does not take effect because the schema file does not exist in the file system.

Example

REFRESH TABLE *dbcarbon.productSalesTable*;

Response

The source Carbon table data is registered with the Hive metabase.

1.7.11 REGISTER INDEX TABLE

Function

This command is used to register an index table with the primary table.

Syntax

REGISTER INDEX TABLE *indextable_name* ON *db_name.maintable_name*;

Parameter Description

Table 1-33 REFRESH INDEX TABLE parameters

Parameter	Description
db_name	Database name. If this parameter is not specified, the current database is selected.
indextable_name	Index table name.
maintable_name	Primary table name.

Precautions

Before running this command, run **REFRESH TABLE** to register the primary table and secondary index table with the Hive metastore.

Examples

create database productdb;

use productdb;

CREATE TABLE productSalesTable(a int,b string,c string) stored as carbondata;

create index productNameIndexTable on table productSalesTable(c) as 'carbondata';

insert into table productSalesTable select 1,'a','aaa';

create database productdb2;

Run the **hdfs** command to copy **productSalesTable** and **productNameIndexTable** in the **productdb** database to the **productdb2** database.

refresh table productdb2.productSalesTable ;

refresh table productdb2.productNameIndexTable ;

explain select * from productdb2.productSalesTable where c = 'aaa'; / The query command does not use an index table.

REGISTER INDEX TABLE productNameIndexTable ON productdb2.productSalesTable;

explain select * from productdb2.productSalesTable where c = 'aaa'; // The query command uses an index table.

System Response

By running this command, the index table will be registered to the primary table.

1.7.12 LOAD DATA

Function

This command is used to load user data of a particular type, so that CarbonData can provide good query performance.

NOTE

Only the raw data on HDFS can be loaded.

Syntax

LOAD DATA *INPATH 'folder_path' INTO TABLE [db_name.]table_name OPTIONS(property_name=property_value, ...);*

Parameter Description

Table 1-34 LOAD DATA parameters

Parameter	Description
folder_path	Path of the file or folder used for storing the raw CSV data.
db_name	Database name. If this parameter is not specified, the current database is used.
table_name	Name of a table in a database.

Precautions

The following configuration items are involved during data loading:

- **DELIMITER**: Delimiters and quote characters provided in the load command. The default value is a comma (**,**).

  OPTIONS('DELIMITER'=',' , 'QUOTECHAR'='"')

  You can use **'DELIMITER'='\t'** to separate CSV data using tabs.

  OPTIONS('DELIMITER'='\t')

  CarbonData also supports **\001** and **\017** as delimiters.

NOTE

When the delimiter of CSV data is a single quotation mark ('), the single quotation mark must be enclosed in double quotation marks (" "). For example, 'DELIMITER'= "'".

- **QUOTECHAR**: Delimiters and quote characters provided in the load command. The default value is double quotation marks (**"**).

  OPTIONS('DELIMITER'=',' , 'QUOTECHAR'='"')

- **COMMENTCHAR**: Comment characters provided in the load command. During data loading, if there is a comment character at the beginning of a line, the line is regarded as a comment line and data in the line will not be loaded. The default value is a pound key (#).

  OPTIONS('COMMENTCHAR'='#')

- **FILEHEADER**: If the source file does not contain any header, add a header to the **LOAD DATA** command.

  OPTIONS('FILEHEADER'='column1,column2')

- **ESCAPECHAR**: Is used to perform strict verification of the escape character on CSV files. The default value is backslash (****).

  OPTIONS('ESCAPECHAR'='\')

  NOTE

  Enter **ESCAPECHAR** in the CSV data. **ESCAPECHAR** must be enclosed in double quotation marks (" "). For example, "a\b".

- Bad records handling:

  In order for the data processing application to provide benefits, certain data integration is required. In most cases, data quality problems are caused by data sources.

  Methods of handling bad records are as follows:

  – Load all of the data before dealing with the errors.

  – Clean or delete bad records before loading data or stop the loading when bad records are found.

  There are many options for clearing source data during CarbonData data loading, as listed in **Table 1-35**.

Table 1-35 Bad Records Logger

Configuration Item	Default Value	Description
BAD_RECORDS_LOGGER_ENABLE	false	Whether to log details about bad records.

Configuration Item	Default Value	Description
BAD_RECORDS_ACTION	FAIL	The four types of actions for bad records are as follows: • **FORCE**: Auto-corrects the data by storing the bad records as NULL. • **REDIRECT**: Bad records cannot be loaded and written to the CSV file in **BAD_RECORD_PATH**. This mode is disabled by default. To enable it, set **carbon.enable.badrecord.action.redirect** to **true**. • **IGNORE**: Bad records are neither loaded nor written to the CSV file. • **FAIL**: Data loading fails if any bad records are found. **NOTE** In loaded data, if all records are bad records, **BAD_RECORDS_ACTION** is invalid and the load operation fails.
IS_EMPTY_DATA_BAD_RECORD	false	Whether empty data of a column to be considered as bad record or not. If this parameter is set to **false**, empty data (""','', or,) is not considered as bad records. If this parameter is set to **true**, empty data is considered as bad records.
BAD_RECORD_PATH	-	HDFS path where bad records are stored. The default value is **Null**. If bad records logging or bad records operation redirection is enabled, the path must be configured by the user.

Example:

LOAD DATA INPATH *'filepath.csv'* **INTO TABLE** *tablename*
OPTIONS('BAD_RECORDS_LOGGER_ENABLE'='true',
'BAD_RECORD_PATH'='hdfs://hacluster/tmp/carbon',
'BAD_RECORDS_ACTION'='REDIRECT',
'IS_EMPTY_DATA_BAD_RECORD'='false');

NOTE

If **REDIRECT** is used, CarbonData will add all bad records into a separate CSV file. However, this file must not be used for subsequent data loading because the content may not exactly match the source record. You must clean up the source record for further data ingestion. This option is used to remind you which records are bad.

- **MAXCOLUMNS**: (Optional) Specifies the maximum number of columns parsed by a CSV parser in a line.

  OPTIONS('MAXCOLUMNS'='400')

Table 1-36 MAXCOLUMNS

Name of the Optional Parameter	Default Value	Maximum Value
MAXCOLUMNS	2000	20000

Table 1-37 Behavior chart of MAXCOLUMNS

MAXCOLUMNS Value	Number of Columns in the File Header	Final Value Considered
Not specified in Load options	5	2000
Not specified in Load options	6000	6000
40	7	Max (column count of file header, MAXCOLUMNS value)
22000	40	20000
60	Not specified in Load options	Max (Number of columns in the first line of the CSV file, MAXCOLUMNS value)

NOTE

There must be sufficient executor memory for setting the maximum value of **MAXCOLUMNS Option**. Otherwise, data loading will fail.

- If **SORT_SCOPE** is set to **GLOBAL_SORT** during table creation, you can specify the number of partitions to be used when sorting data. If this parameter is not set or is set to a value less than **1**, the number of map tasks is used as the number of reduce tasks. It is recommended that each reduce task process 512 MB to 1 GB data.

  OPTIONS('GLOBAL_SORT_PARTITIONS'='2')

NOTE

To increase the number of partitions, you may need to increase the value of **spark.driver.maxResultSize**, as the sampling data collected in the driver increases with the number of partitions.

- **DATEFORMAT**: Specifies the date format of the table.

  OPTIONS('DATEFORMAT'='dateFormat')

  NOTE

  Date formats are specified by date pattern strings. The date pattern letters in Carbon are same as in JAVA.

- **TIMESTAMPFORMAT**: Specifies the timestamp of a table.
- *OPTIONS('TIMESTAMPFORMAT'='timestampFormat')*
- **SKIP_EMPTY_LINE**: Ignores empty rows in the CSV file during data loading.

  OPTIONS('SKIP_EMPTY_LINE'='TRUE/FALSE')

- **Optional: SCALE_FACTOR**: Used to control the number of partitions for **RANGE_COLUMN**, **SCALE_FACTOR**. The formula is as follows:

```
splitSize = max(blocklet_size, (block_size - blocklet_size)) * scale_factor
numPartitions = total size of input data / splitSize
```

  The default value is **3**. The value ranges from **1** to **300**.

  OPTIONS('SCALE_FACTOR'='10')

  NOTE

  - If **GLOBAL_SORT_PARTITIONS** and **SCALE_FACTOR** are used at the same time, only **GLOBAL_SORT_PARTITIONS** is valid.
  - The compaction on **RANGE_COLUMN** will use **LOCAL_SORT** by default.

Scenarios

To load a CSV file to a CarbonData table, run the following statement:

LOAD DATA *INPATH 'folder path' INTO TABLE tablename OPTIONS(property_name=property_value, ...);*

Examples

The data in the **data.csv** file is as follows:

```
ID,date,country,name,phonetype,serialname,salary
4,2014-01-21 00:00:00,xxx,aaa4,phone2435,ASD66902,15003
5,2014-01-22 00:00:00,xxx,aaa5,phone2441,ASD90633,15004
6,2014-03-07 00:00:00,xxx,aaa6,phone294,ASD59961,15005
```

CREATE TABLE carbontable(ID int, date Timestamp, country String, name String, phonetype String, serialname String,salary int) STORED AS carbondata;

LOAD DATA *inpath 'hdfs://hacluster/tmp/data.csv' INTO table carbontable*

options('DELIMITER'=',');

System Response

Success or failure will be recorded in the driver logs.

1.7.13 UPDATE CARBON TABLE

Function

This command is used to update the CarbonData table based on the column expression and optional filtering conditions.

Syntax

- Syntax 1:

  UPDATE <CARBON TABLE> SET (column_name1, column_name2, ... column_name n) = (column1_expression , column2_expression , column3_expression ... column n_expression) [WHERE { <filter_condition> }];

- Syntax 2:

  UPDATE <CARBON TABLE> SET (column_name1, column_name2,) = (select sourceColumn1, sourceColumn2 from sourceTable [WHERE { <filter_condition> }]) [WHERE { <filter_condition> }];

Parameter Description

Table 1-38 UPDATE parameters

Parameter	Description
CARBON TABLE	Name of the CarbonData table to be updated
column_name	Target column to be updated
sourceColumn	Column value of the source table that needs to be updated in the target table
sourceTable	Table from which the records are updated to the target table

Precautions

Note the following before running this command:

- The UPDATE command fails if multiple input rows in the source table are matched with a single row in the target table.
- If the source table generates empty records, the UPDATE operation completes without updating the table.
- If rows in the source table do not match any existing rows in the target table, the UPDATE operation completes without updating the table.
- UPDATE is not allowed in the table with secondary index.
- In a subquery, if the source table and target table are the same, the UPDATE operation fails.
- The UPDATE operation fails if the subquery used in the UPDATE command contains an aggregate function or a GROUP BY clause.

For example, **update t_carbn01 a set (a.item_type_code, a.profit) = (select b.item_type_cd, sum(b.profit) from t_carbn01b b where item_type_cd =2 group by item_type_code);**.

In the preceding example, aggregate function **sum(b.profit)** and GROUP BY clause are used in the subquery. As a result, the UPDATE operation will fail.

- If the **carbon.input.segments** property has been set for the queried table, the UPDATE operation fails. To solve this problem, run the following statement before the query:

  Syntax:

  SET carbon.input.segments. <database_name>. <table_name>=*;

Examples

- Example 1:

  update carbonTable1 d set (d.column3,d.column5) = (select s.c33 ,s.c55 from sourceTable1 s where d.column1 = s.c11) where d.column1 = 'country' exists(select * from table3 o where o.c2 > 1);

- Example 2:

  update carbonTable1 d set (c3) = (select s.c33 from sourceTable1 s where d.column1 = s.c11) where exists(select * from iud.other o where o.c2 > 1);

- Example 3:

  update carbonTable1 set (c2, c5) = (c2 + 1, concat(c5 , "y"));

- Example 4:

  update carbonTable1 d set (c2, c5) = (c2 + 1, "xyx") where d.column1 = 'india';

- Example 5:

  update carbonTable1 d set (c2, c5) = (c2 + 1, "xyx") where d.column1 = 'india' and exists(select * from table3 o where o.column2 > 1);

System Response

Success or failure will be recorded in the driver log and on the client.

1.7.14 DELETE RECORDS from CARBON TABLE

Function

This command is used to delete records from a CarbonData table.

Syntax

DELETE FROM CARBON_TABLE [WHERE expression];

Parameter Description

Table 1-39 DELETE RECORDS parameters

Parameter	Description
CARBON TABLE	Name of the CarbonData table in which the DELETE operation is performed

Precautions

- If a segment is deleted, all secondary indexes associated with the segment are deleted as well.
- If the **carbon.input.segments** property has been set for the queried table, the DELETE operation fails. To solve this problem, run the following statement before the query:

  Syntax:

  SET carbon.input.segments. <database_name>.<table_name>=*;

Examples

- Example 1:

  delete from columncarbonTable1 d where d.column1 = 'country';
- Example 2:

  delete from dest where column1 IN ('country1', 'country2');
- Example 3:

  delete from columncarbonTable1 where column1 IN (select column11 from sourceTable2);
- Example 4:

  delete from columncarbonTable1 where column1 IN (select column11 from sourceTable2 where column1 = 'xxx');
- Example 5:

  delete from columncarbonTable1 where column2 >= 4;

System Response

Success or failure will be recorded in the driver log and on the client.

1.7.15 INSERT INTO CARBON TABLE

Function

This command is used to insert the output of the SELECT command to a Carbon table.

Syntax

INSERT INTO [CARBON TABLE] [select query];

Parameter Description

Table 1-40 INSERT INTO parameters

Parameter	Description
CARBON TABLE	Name of the CarbonData table where data is to be inserted
select query	SELECT query on the source table (CarbonData, Hive, and Parquet tables are supported)

Precautions

- A table has been created.
- You must belong to the data loading group in order to perform data loading operations. By default, the data loading group is named **ficommon**.
- CarbonData tables cannot be overwritten.
- The data type of the source table and the target table must be the same. Otherwise, data in the source table will be regarded as bad records.
- The **INSERT INTO** command does not support partial success. If bad records exist, the command fails.
- When you insert data of the source table to the target table, you cannot upload or update data of the source table.

  To enable data loading or updating during the INSERT operation, set the following parameter to **true**.

  carbon.insert.persist.enable=**true**

  By default, the preceding parameters are set to **false**.

  NOTE

  Enabling this property will reduce the performance of the INSERT operation.

Example

create table carbon01(a int,b string,c string) stored as carbondata;

insert into table carbon01 values(1,'a','aa'),(2,'b','bb'),(3,'c','cc');

create table carbon02(a int,b string,c string) stored as carbondata;

INSERT INTO carbon02 select * from carbon01 where a > 1;

System Response

Success or failure will be recorded in the driver logs.

1.7.16 DELETE SEGMENT by ID

Function

This command is used to delete a segment based on the segment ID.

Syntax

DELETE FROM TABLE db_name.table_name WHERE SEGMENT.ID IN (segment_id1,segment_id2);

Parameters Description

Table 1-41 DELETE LOAD parameters

Parameter	Description
segment_id	ID of the segment to be deleted.
db_name	Database name. If this parameter is not specified, the current database is used.
table_name	Name of a table in the specified database.

Precautions

Segments cannot be deleted from stream tables.

Example

DELETE FROM TABLE CarbonDatabase.CarbonTable WHERE SEGMENT.ID IN (0);

DELETE FROM TABLE CarbonDatabase.CarbonTable WHERE SEGMENT.ID IN (0,5,8);

Response

Success or failure will be recorded in CarbonData logs.

1.7.17 DELETE SEGMENT by DATE

Function

This command is used to delete segments by loading date. Segments created before a specific date will be deleted.

Syntax

DELETE FROM TABLE db_name.table_name WHERE SEGMENT.STARTTIME BEFORE date_value;

Parameter Description

Table 1-42 DELETE SEGMENT by DATE parameters

Parameter	Description
db_name	Database name. If this parameter is not specified, the current database is used.
table_name	Name of a table in the specified database
date_value	Valid date when segments are started to be loaded. Segments before the date will be deleted.

Precautions

Segments cannot be deleted from the stream table.

Example

DELETE FROM TABLE db_name.table_name WHERE SEGMENT.STARTTIME BEFORE '2017-07-01 12:07:20';

STARTTIME indicates the loading start time of different loads.

System Response

Success or failure will be recorded in CarbonData logs.

1.7.18 SHOW SEGMENTS

Function

This command is used to list the segments of a CarbonData table.

Syntax

SHOW SEGMENTS FOR TABLE *[db_name.]table_name* **LIMIT** *number_of_loads;*

Parameter Description

Table 1-43 SHOW SEGMENTS FOR TABLE parameters

Parameter	Description
db_name	Database name. If this parameter is not specified, the current database is used.
table_name	Name of a table in the specified database
number_of_loads	Threshold of records to be listed

Precautions

None

Examples

create table carbon01(a int,b string,c string) stored as carbondata;

insert into table carbon01 select 1,'a','aa';

insert into table carbon01 select 2,'b','bb';

insert into table carbon01 select 3,'c','cc';

SHOW SEGMENTS FOR TABLE carbon01 LIMIT 2;

System Response

```
+-----+----------+--------------------------+------------------+------------+------------+-------------+--------------+--
+
| ID  |  Status  |     Load Start Time      | Load Time Taken  | Partition  | Data Size  | Index Size  | File Format  |
+-----+----------+--------------------------+------------------+------------+------------+-------------+--------------+--
+
| 3   | Success  | 2020-09-28 22:53:26.336  | 3.726S           | {}         | 6.47KB     | 3.30KB      | columnar_v3  |
| 2   | Success  | 2020-09-28 22:53:01.702  | 6.688S           | {}         | 6.47KB     | 3.30KB      | columnar_v3  |
+-----+----------+--------------------------+------------------+------------+------------+-------------+--------------+--
+
```

1.7.19 CREATE SECONDARY INDEX

Function

This command is used to create secondary indexes in the CarbonData tables.

Syntax

CREATE INDEX *index_name*

ON TABLE *[db_name.]table_name (col_name1, col_name2)*

AS *'carbondata'*

PROPERTIES *('table_blocksize'='256')*;

Parameter Description

Table 1-44 CREATE SECONDARY INDEX parameters

Parameter	Description
index_name	Index table name. It consists of letters, digits, and special characters (_).
db_name	Database name. It consists of letters, digits, and special characters (_).
table_name	Name of the database table. It consists of letters, digits, and special characters (_).

Parameter	Description
col_name	Name of a column in a table. Multiple columns are supported. It consists of letters, digits, and special characters (_).
table_blocksize	Block size of a data file. For details, see •**Block Size**.

Precautions

db_name is optional.

Examples

create table productdb.productSalesTable(id int,price int,productName string,city string) stored as carbondata;

CREATE INDEX productNameIndexTable on table productdb.productSalesTable (productName,city) as 'carbondata' ;

In this example, a secondary table named **productdb.productNameIndexTable** is created and index information of the provided column is loaded.

System Response

A secondary index table will be created. Index information related to the provided column will be loaded into the secondary index table. The success message will be recorded in system logs.

1.7.20 SHOW SECONDARY INDEXES

Function

This command is used to list all secondary index tables in the CarbonData table.

Syntax

SHOW INDEXES ON db_name.table_name;

Parameter Description

Table 1-45 SHOW SECONDARY INDEXES parameters

Parameter	Description
db_name	Database name. It consists of letters, digits, and special characters (_).
table_name	Name of the database table. It consists of letters, digits, and special characters (_).

Precautions

db_name is optional.

Examples

create table productdb.productSalesTable(id int,price int,productName string,city string) stored as carbondata;

CREATE INDEX productNameIndexTable on table productdb.productSalesTable (productName,city) as 'carbondata' ;

SHOW INDEXES ON productdb.productSalesTable;

System Response

All index tables and corresponding index columns in a given CarbonData table will be listed.

1.7.21 DROP SECONDARY INDEX

Function

This command is used to delete a secondary index table from a specified table.

Syntax

DROP INDEX [IF EXISTS] index_name ON [db_name.]table_name;

Parameter Description

Table 1-46 Parameter parameters

Parameter	Description
index_name	Index table name. It consists of letters, digits, and special characters (_).
db_name	Database name. If this parameter is not specified, the current default database is selected.
table_name	Name of the table to be deleted.

Precautions

IF EXISTS and db_name are optional.

Example

DROP INDEX if exists productNameIndexTable ON productdb.productSalesTable;

Response

The secondary index table is deleted, and the index information is deleted from the CarbonData table. The deletion success message is recorded in the system log.

1.7.22 CLEAN FILES

Function

After the **DELETE SEGMENT** command is executed, the deleted segments are marked as the **delete** state. After the segments are merged, the status of the original segments changes to **compacted**. The data files of these segments are not physically deleted. If you want to forcibly delete these files, run the **CLEAN FILES** command.

However, running this command may result in a query command execution failure.

Syntax

CLEAN FILES FOR TABLE *[db_name.]table_name* ;

Parameter Description

Table 1-47 CLEAN FILES FOR TABLE parameters

Parameter	Description
db_name	Database name. It consists of letters, digits, and underscores (_).
table_name	Name of the database table. It consists of letters, digits, and underscores (_).

Precautions

None

Examples

Add Carbon configuration parameters.

```
carbon.clean.file.force.allowed = true
```

create table carbon01(a int,b string,c string) stored as carbondata;

insert into table carbon01 select 1,'a','aa';

insert into table carbon01 select 2,'b','bb';

delete from table carbon01 where segment.id in (0);

show segments for table carbon01;

CLEAN FILES FOR TABLE carbon01 options('force'='true');

show segments for table carbon01;

In this example, all the segments marked as **deleted** and **compacted** are physically deleted.

System Response

Success or failure will be recorded in the driver logs.

1.7.23 SET/RESET

Function

This command is used to dynamically add, update, display, or reset the CarbonData properties without restarting the driver.

Syntax

- Add or Update parameter value:

  SET *parameter_name=parameter_value*

  This command is used to add or update the value of **parameter_name**.
- Display property value:

  SET *parameter_name*

  This command is used to display the value of **parameter_name**.
- Display session parameter:

  SET

  This command is used to display all supported session parameters.
- Display session parameters along with usage details:

  SET -v

  This command is used to display all supported session parameters and their usage details.
- Reset parameter value:

  RESET

  This command is used to clear all session parameters.

Parameter Description

Table 1-48 SET parameters

Parameter	Description
parameter_name	Name of the parameter whose value needs to be dynamically added, updated, or displayed
parameter_value	New value of **parameter_name** to be set

Precautions

The following table lists the properties which you can set or clear using the SET or RESET command.

Table 1-49 Properties

Property	Description
carbon.options.bad.records.logger.enable	Whether to enable bad record logger.
carbon.options.bad.records.action	Operations on bad records, for example, force, redirect, fail, or ignore. For more information, see **•Bad record handling**.
carbon.options.is.empty.data.bad.record	Whether the empty data is considered as a bad record. For more information, see **Bad record handling**.
carbon.options.sort.scope	Scope of the sort during data loading.
carbon.options.bad.record.path	HDFS path where bad records are stored.
carbon.custom.block.distribution	Whether to enable Spark or CarbonData block distribution.
enable.unsafe.sort	Whether to use unsafe sort during data loading. Unsafe sort reduces the garbage collection during data loading, thereby achieving better performance.
carbon.si.lookup.partialstring	If this is set to **TRUE**, the secondary index uses the starts-with, ends-with, contains, and LIKE partition condition strings. If this is set to **FALSE**, the secondary index uses only the starts-with partition condition string.

Property	Description
carbon.input.segments	Segment ID to be queried. This property allows you to query a specified segment of a specified table. CarbonScan reads data only from the specified segment ID. Syntax: **carbon.input.segments.<database_name>.<table_name> = <list of segment ids >** If you want to query a specified segment in multi-thread mode, you can use **CarbonSession.threadSet** instead of the **SET** statement. Syntax: **CarbonSession.threadSet ("carbon.input.segments.<database_name>.<table_name>","<list of segment ids >");** **NOTE** You are advised not to set this property in the **carbon.properties** file because all sessions contain the segment list unless session-level or thread-level overwriting occurs.

Examples

- Add or Update:

  SET *enable.unsafe.sort=true*

- Display property value:

  SET *enable.unsafe.sort*

- Show the segment ID list, segment status, and other required details, and specify the segment list to be read:

  SHOW SEGMENTS FOR *TABLE carbontable1;*

  SET *carbon.input.segments.db.carbontable1 = 1, 3, 9;*

- Query a specified segment in multi-thread mode:

  CarbonSession.threadSet (*"carbon.input.segments.default.carbon_table_MulTI_THread", "1,3"*);

- Use **CarbonSession.threadSet** to query segments in a multi-thread environment (Scala code is used as an example):

```
def main(args: Array[String]) {
 Future
{             CarbonSession.threadSet("carbon.input.segments.default.carbon_table_MulTI_THread", "1")
     spark.sql("select count(empno) from carbon_table_MulTI_THread").show()
   }
}
```


- Reset:
  RESET

System Response

- You can view the success result in driver logs.
- You can view the failure result on the UI.

1.7.24 Concurrent CarbonData Table Operations

Before performing DDL and DML operations, you need to obtain the corresponding locks. See **Table 1-50** for details about the locks that need to be obtained for each operation. The check mark (√) indicates that the lock is required. An operation can be performed only after all required locks are obtained.

You can check whether any two operations can be executed concurrently by using the following method: The first two lines in **Table 1-50** indicate two operations. If no column in the two lines is marked with the check mark (√), the two operations can be executed concurrently. That is, if the columns with check marks (√) in the two lines do not exist, the two operations can be executed concurrently.

Table 1-50 List of obtaining locks for operations

Operation	METADATA_LOCK	COMPACTION_LOCK	DROP_TABLE_LOCK	DELETE_SEGMENT_LOCK	CLEAN_FILES_LOCK	ALTER_PARTITION_LOCK	UPDATE_LOCK	STREAMING_LOCK	CONCURRENT_LOAD_LOCK	SEGMENT_LOCK
CREATE TABLE	-	-	-	-	-	-	-	-	-	-
CREATE TABLE As SELECT	-	-	-	-	-	-	-	-	-	-
DROP TABLE	√	-	√	-	-	-	-	√	-	-
ALTER TABLE COMPACTION	-	√	-	-	-	-	√	-	-	-

Operation	METADATA_LOCK	COMPACTION_LOCK	DROP_TABLE_LOCK	DELETE_SEGMENT_LOCK	CLEAN_FILES_LOCK	ALTER_PARTITION_LOCK	UPDATE_LOCK	STREAMING_LOCK	CONCURRENT_LOAD_LOCK	SEGMENT_LOCK
TABLE RENAME	-	-	-	-	-	-	-	-	-	-
ADD COLUMNS	√	√	-	-	-	-	-	-	-	-
DROP COLUMNS	√	√	-	-	-	-	-	-	-	-
CHANGE DATA TYPE	√	√	-	-	-	-	-	-	-	-
REFRESH TABLE	-	-	-	-	-	-	-	-	-	-
REGISTER INDEX TABLE	√	-	-	-	-	-	-	-	-	-
REFRESH INDEX	-	√	-	-	-	-	-	-	-	-
LOAD DATA/ INSERT INTO	-	-	-	-	-	-	-	-	√	√

Operation	METADATA_LOCK	COMPACTION_LOCK	DROP_TABLE_LOCK	DELETE_SEGMENT_LOCK	CLEAN_FILES_LOCK	ALTER_PARTITION_LOCK	UPDATE_LOCK	STREAMING_LOCK	CONCURRENT_LOAD_LOCK	SEGMENT_LOCK
UPDATE CARBON TABLE	√	√	-	-	-	-	√	-	-	-
DELETE RECORDS from CARBON TABLE	√	√	-	-	-	-	√	-	-	-
DELETE SEGMENT by ID	-	-	-	√	√	-	-	-	-	-
DELETE SEGMENT by DATE	-	-	-	√	√	-	-	-	-	-
SHOW SEGMENTS	-	-	-	-	-	-	-	-	-	-
CREATE SECONDARY INDEX	√	√	-	√	-	-	-	-	-	-

Operation	METADATA_LOCK	COMPACTION_LOCK	DROP_TABLE_LOCK	DELETE_SEGMENT_LOCK	CLEAN_FILES_LOCK	ALTER_PARTITION_LOCK	UPDATE_LOCK	STREAMING_LOCK	CONCURRENT_LOAD_LOCK	SEGMENT_LOCK
SHOW SECONDARY INDEXES	-	-	-	-	-	-	-	-	-	-
DROP SECONDARY INDEX	√	-	√	-	-	-	-	-	-	-
CLEAN FILES	-	-	-	-	-	-	-	-	-	-
SET/RESET	-	-	-	-	-	-	-	-	-	-
Add Hive Partition	-	-	-	-	-	-	-	-	-	-
Drop Hive Partition	√	√	√	√	√	√	-	-	-	-
Drop Partition	√	√	√	√	√	√	-	-	-	-
Alter table set	√	√	-	-	-	-	-	-	-	-

1.7.25 CarbonData Segment API

This topic describes the Segment APIs and their usage. All methods are in the **org.apache.spark.util.CarbonSegmentUtil** class.

The following method has been discarded:

```
/**
* Returns the valid segments for the query based on the filter condition
* present in carbonScanRdd.
*
* @param carbonScanRdd
* @return Array of valid segments
*/
@deprecated def getFilteredSegments(carbonScanRdd: CarbonScanRDD[InternalRow]): Array[String];
```

How to Use

Use the following method to obtain CarbonScanRDD from the query statement:

```
val df=carbon.sql("select * from table where age='12'")
val myscan=df.queryExecution.sparkPlan.collect {
case scan: CarbonDataSourceScan if scan.rdd.isInstanceOf[CarbonScanRDD[InternalRow]] => scan.rdd
case scan: RowDataSourceScanExec if scan.rdd.isInstanceOf[CarbonScanRDD[InternalRow]] => scan.rdd
}.head
val carbonrdd=myscan.asInstanceOf[CarbonScanRDD[InternalRow]]
```

The following is an example:

```
CarbonSegmentUtil.getFilteredSegments(carbonrdd)
```

You can pass the following SQL statement to obtain the filtered segment:

```
/**
* Returns an array of valid segment numbers based on the filter condition provided in the sql
* NOTE: This API is supported only for SELECT Sql (insert into,ctas,.., is not supported)
*
* @param sql
* @param sparkSession
* @return Array of valid segments
* @throws UnsupportedOperationException because Get Filter Segments API supports if and only
* if only one carbon main table is present in query.
*/
def getFilteredSegments(sql: String, sparkSession: SparkSession): Array[String];
```

The following is an example:

```
CarbonSegmentUtil.getFilteredSegments("select * from table where age='12'", sparkSession)
```

Pass the database name and table name to obtain the list of segments to be merged. The obtained segment list can be used as the parameter of the **getMergedLoadName** function.

```
/**
* Identifies all segments which can be merged with MAJOR compaction type.
* NOTE: This result can be passed to getMergedLoadName API to get the merged load name.
*
* @param sparkSession
* @param tableName
* @param dbName
* @return list of LoadMetadataDetails
*/
def identifySegmentsToBeMerged(sparkSession: SparkSession,
tableName: String,
dbName: String) : util.List[LoadMetadataDetails];
```

The following is an example:

```
CarbonSegmentUtil.identifySegmentsToBeMerged(sparkSession, "table_test","default")
```

Pass the database name, table name, and custom segment list to obtain the list of segments that will be merged on demand. The obtained segment list can be used as the parameter of the **getMergedLoadName** function.


```
/**
* Identifies all segments which can be merged with CUSTOM compaction type.
*  NOTE: This result can be passed to getMergedLoadName API to get the merged load name.
*
* @param sparkSession
* @param tableName
* @param dbName
* @param customSegments
* @return list of LoadMetadataDetails
* @throws UnsupportedOperationException if customSegments is null or empty.
* @throws MalformedCarbonCommandException if segment does not exist or is not valid
*/
def identifySegmentsToBeMergedCustom(sparkSession: SparkSession,
tableName: String,
dbName: String,
customSegments: util.List[String]): util.List[LoadMetadataDetails];
```

The following is an example:

```
val customSegments = new util.ArrayList[String]()
customSegments.add("1")
customSegments.add("2")
CarbonSegmentUtil.identifySegmentsToBeMergedCustom(sparkSession, "table_test","default",
customSegments)
```

Pass a specified segment list, and return the new Merged Load Name.

```
/**
* Returns the Merged Load Name for given list of segments
*
* @param list of segments
* @return Merged Load Name
* @throws UnsupportedOperationException if list of segments is less than 1
*/
def getMergedLoadName(list: util.List[LoadMetadataDetails]): String;
```

The following is an example:

```
val carbonTable = CarbonEnv.getCarbonTable(Option(databaseName), tableName)(sparkSession)
val loadMetadataDetails = SegmentStatusManager.readLoadMetadata(carbonTable.getMetadataPath)
CarbonSegmentUtil.getMergedLoadName(loadMetadataDetails.toList.asJava)
```

1.7.26 CarbonData Tablespace Index

Quick Example

```
create table IF NOT EXISTS carbonTable
(
COLUMN1    BIGINT,
LONGITUDE    BIGINT,
LATITUDE    BIGINT,
COLUMN2    BIGINT,
COLUMN3    BIGINT
)
STORED AS carbondata
TBLPROPERTIES
('SPATIAL_INDEX.mygeohash.type'='geohash','SPATIAL_INDEX.mygeohash.sourcecolumns'='longitude,
latitude','SPATIAL_INDEX.mygeohash.originLatitude'='39.850713','SPATIAL_INDEX.mygeohash.gridSize'='50','S
PATIAL_INDEX.mygeohash.minLongitude'='115.828503','SPATIAL_INDEX.mygeohash.maxLongitude'='720.000
000','SPATIAL_INDEX.mygeohash.minLatitude'='39.850713','SPATIAL_INDEX.mygeohash.maxLatitude'='720.0
00000','SPATIAL_INDEX'='mygeohash','SPATIAL_INDEX.mygeohash.conversionRatio'='1000000','SORT_COLU
MNS'='column1,column2,column3,latitude,longitude');
```

Introduction to Spatial Indexes

Spatial data includes multidimensional points, lines, rectangles, cubes, polygons, and other geometric objects. A spatial data object occupies a certain region of

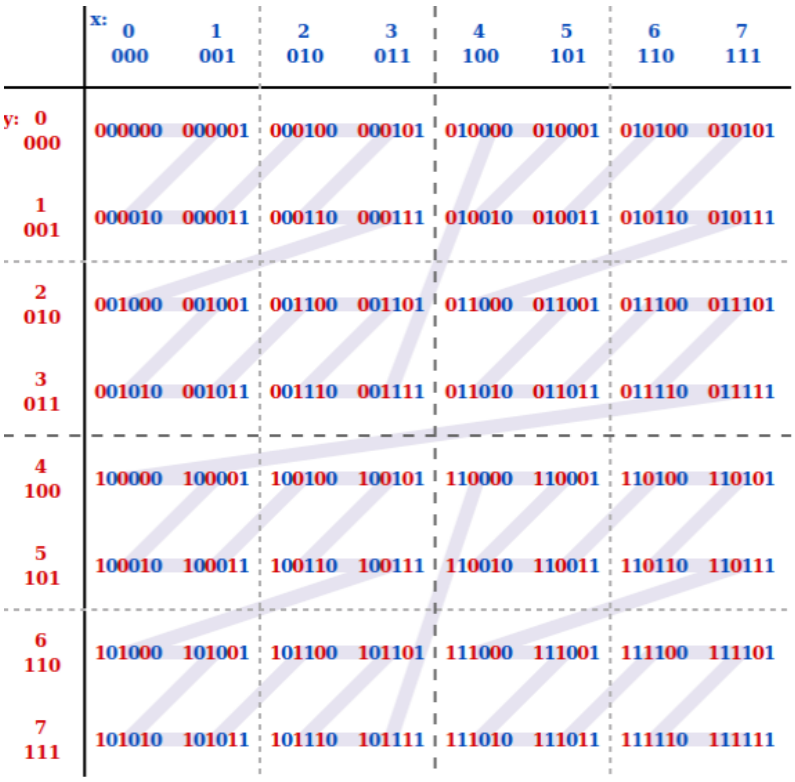
space, called spatial scope, characterized by its location and boundary. The spatial data can be either point data or region data.

- Point data: A point has a spatial extent characterized completely by its location. It does not occupy space and has no associated boundary. Point data consists of a collection of points in a two-dimensional space. Points can be stored as a pair of longitude and latitude.
- Region data: A region has a spatial extent with a location, and boundary. The location can be considered as the position of a fixed point in the region, such as its centroid. In two dimensions, the boundary can be visualized as a line (for finite regions, a closed loop). Region data contains a collection of regions.

Currently, only point data is supported, and it can be stored.

Longitude and latitude can be encoded as a unique GeoID. Geohash is a public-domain geocoding system invented by Gustavo Niemeyer. It encodes geographical locations into a short string of letters and digits. It is a hierarchical spatial data structure which subdivides the space into buckets of grid shape, which is one of the many applications of what is known as the Z-order curve, and generally the space-filling curve.

The Z value of a point in multiple dimensions is calculated by interleaving the binary representation of its coordinate value, as shown in the following figure. When Geohash is used to create a GeoID, data is sorted by GeoID instead of longitude and latitude. Data is stored by spatial proximity.



Creating a Table

GeoHash encoding:

```
create table IF NOT EXISTS carbonTable
(
```



```
...
`LONGITUDE`     BIGINT,
`LATITUDE`      BIGINT,
...
)
STORED AS carbondata
TBLPROPERTIES
('SPATIAL_INDEX.mygeohash.type'='geohash','SPATIAL_INDEX.mygeohash.sourcecolumns'='longitude,
latitude','SPATIAL_INDEX.mygeohash.originLatitude'='xx.xxxxxx','SPATIAL_INDEX.mygeohash.gridSize'='xx','SP
ATIAL_INDEX.mygeohash.minLongitude'='xxx.xxxxxx','SPATIAL_INDEX.mygeohash.maxLongitude'='xxx.xxxxxx'
,'SPATIAL_INDEX.mygeohash.minLatitude'='xx.xxxxxx','SPATIAL_INDEX.mygeohash.maxLatitude'='xxx.xxxxxx','
SPATIAL_INDEX'='mygeohash','SPATIAL_INDEX.mygeohash.conversionRatio'='1000000','SORT_COLUMNS'='co
lumn1,column2,column3,latitude,longitude');
```

SPATIAL_INDEX is a user-defined index handler. This handler allows users to create new columns from the table-structure column set. The new column name is the same as that of the handler name. The **type** and **sourcecolumns** properties of the handler are mandatory. Currently, the value of **type** supports only **geohash**. Carbon provides a default implementation class that can be easily used. You can extend the default implementation class to mount the customized implementation class of **geohash**. The default handler also needs to provide the following table properties:

- **SPATIAL_INDEX.*xxx*.originLatitude**: specifies the origin latitude. (**Double** type.)
- **SPATIAL_INDEX.*xxx*.gridSize**: specifies the grid length in meters. (**Int** type.)
- **SPATIAL_INDEX.*xxx*.minLongitude**: specifies the minimum longitude. (**Double** type.)
- **SPATIAL_INDEX.*xxx*.maxLongitude**: specifies the maximum longitude. (**Double** type.)
- **SPATIAL_INDEX.*xxx*.minLatitude**: specifies the minimum latitude. (**Double** type.)
- **SPATIAL_INDEX.*xxx*.maxLatitude**: specifies the maximum latitude. (**Double** type.)
- **SPATIAL_INDEX.*xxx*.conversionRatio**: used to convert the small value of the longitude and latitude to an integer. (**Int** type.)

You can add your own table properties to the handlers in the above format and access them in your custom implementation class. **originLatitude**, **gridSize**, and **conversionRatio** are mandatory. Other parameters are optional in Carbon. You can use the **SPATIAL_INDEX.*xxx*.class** property to specify their implementation classes.

The default implementation class can generate handler column values for **sourcecolumns** in each row and support query based on the **sourcecolumns** filter criteria. The generated handler column is invisible to users. Except the **SORT_COLUMNS** table properties, no DDL commands or properties are allowed to contain the handler column.

NOTE

- By default, the generated handler column is regarded as the sorting column. If **SORT_COLUMNS** does not contain any **sourcecolumns**, add the handler column to the end of the existing **SORT_COLUMNS**. If the handler column has been specified in **SORT_COLUMNS**, its order in **SORT_COLUMNS** remains unchanged.
- If **SORT_COLUMNS** contains any **sourcecolumns** but does not contain the handler column, the handler column is automatically inserted before **sourcecolumns** in **SORT_COLUMNS**.
- If **SORT_COLUMNS** needs to contain any **sourcecolumns**, ensure that the handler column is listed before the **sourcecolumns** so that the handler column can take effect during sorting.

GeoSOT encoding:

```
CREATE TABLE carbontable(
...
longitude DOUBLE,
latitude DOUBLE,
...)
STORED AS carbondata
TBLPROPERTIES ('SPATIAL_INDEX'='xxx',
'SPATIAL_INDEX.xxx.type'='geosot',
'SPATIAL_INDEX.xxx.sourcecolumns'='longitude, latitude',
'SPATIAL_INDEX.xxx.level'='21',
'SPATIAL_INDEX.xxx.class'='org.apache.carbondata.geo.GeoSOTIndex')
```

Table 1-51 Parameter description

Parameter	Description
SPATIAL_INDEX	Specifies the spatial index. Its value is the same as the column name.
SPATIAL_INDEX.xxx.type	(Mandatory) The value is **geosot**.
SPATIAL_INDEX.xxx.source columns	(Mandatory) Specifies the source columns for calculating the spatial index. The value must be two existing columns of the double type.
SPATIAL_INDEX.xxx.level	(Optional) Specifies the columns for calculating the spatial index. The default value is **17**, through which you can obtain an accurate result and improve the computing performance.
SPATIAL_INDEX.xxx.class	(Optional) Specifies the implementation class of GeoSOT. The default value is **org.apache.carbondata.geo.GeoSOTIndex**.

Example:

```
create table geosot(
timevalue bigint,
longitude double,
latitude double)
stored as carbondata
TBLPROPERTIES ('SPATIAL_INDEX'='mygeosot',
'SPATIAL_INDEX.mygeosot.type'='geosot',
'SPATIAL_INDEX.mygeosot.level'='21', 'SPATIAL_INDEX.mygeosot.sourcecolumns'='longitude, latitude');
```


Preparing Data

- Data file 1: **geosotdata.csv**

```
timevalue,longitude,latitude
1575428400000,116.285807,40.084087
1575428400000,116.372142,40.129503
1575428400000,116.187332,39.979316
1575428400000,116.337069,39.951887
1575428400000,116.359102,40.154684
1575428400000,116.736367,39.970323
1575428400000,116.720179,40.009893
1575428400000,116.346961,40.13355
1575428400000,116.302895,39.930753
1575428400000,116.288955,39.999101
1575428400000,116.17609,40.129953
1575428400000,116.725575,39.981115
1575428400000,116.266922,40.179415
1575428400000,116.353706,40.156483
1575428400000,116.362699,39.942444
1575428400000,116.325378,39.963129
```

- Data file 2: **geosotdata2.csv**

```
timevalue,longitude,latitude
1575428400000,120.17708,30.326882
1575428400000,120.180685,30.326327
1575428400000,120.184976,30.327105
1575428400000,120.189311,30.327549
1575428400000,120.19446,30.329698
1575428400000,120.186965,30.329133
1575428400000,120.177481,30.328911
1575428400000,120.169713,30.325614
1575428400000,120.164563,30.322243
1575428400000,120.171558,30.319613
1575428400000,120.176365,30.320687
1575428400000,120.179669,30.323688
1575428400000,120.181001,30.320761
1575428400000,120.187094,30.32354
1575428400000,120.193574,30.323651
1575428400000,120.186192,30.320132
1575428400000,120.190055,30.317464
1575428400000,120.195376,30.318094
1575428400000,120.160786,30.317094
1575428400000,120.168211,30.318057
1575428400000,120.173618,30.316612
1575428400000,120.181001,30.317316
1575428400000,120.185162,30.315908
1575428400000,120.192415,30.315871
1575428400000,120.161902,30.325614
1575428400000,120.164306,30.328096
1575428400000,120.197093,30.325985
1575428400000,120.19602,30.321651
1575428400000,120.198638,30.32354
1575428400000,120.165421,30.314834
```

Importing Data

The GeoHash default implementation class extends the customized index abstract class. If the handler property is not set to a customized implementation class, the default implementation class is used. You can extend the default implementation class to mount the customized implementation class of **geohash**. The methods of the customized index abstract class are as follows:

- **Init** method: Used to extract, verify, and store the handler property. If the operation fails, the system throws an exception and displays the error information.

- **Generate** method: Used to generate indexes. It generates an index for each row of data.
- **Query** method: Used to generate an index value range list for given input.

The commands for importing data are the same as those for importing common Carbon tables.

LOAD DATA inpath '/tmp/*geosotdata.csv*' INTO TABLE geosot OPTIONS ('DELIMITER'= ',');

LOAD DATA inpath '/tmp/*geosotdata2.csv*' INTO TABLE geosot OPTIONS ('DELIMITER'= ',');

NOTE

For details about **geosotdata.csv** and **geosotdata2.csv**, see **Preparing Data**.

Aggregate Query of Irregular Spatial Sets

Query statements and filter UDFs

- Filtering data based on polygon

  IN_POLYGON(pointList)

  UDF input parameter

Parameter	Type	Description
pointList	String	Enter multiple points as a string. Each point is presented as **longitude latitude**. Longitude and latitude are separated by a space. Each pair of longitude and latitude is separated by a comma (,). The longitude and latitude values at the start and end of the string must be the same.

  UDF output parameter

Parameter	Type	Description
inOrNot	Boolean	Checks whether data is in the specified **polygon_list**.

  Example:

```
select longitude, latitude from geosot where IN_POLYGON('116.321011 40.123503, 116.137676 39.947911, 116.560993 39.935276, 116.321011 40.123503');
```

- Filtering data based on the polygon list

  IN_POLYGON_LIST(polygonList, opType)

  UDF input parameters

Parameter	Type	Description
polygonList	String	Inputs multiple polygons as a string. Each polygon is presented as **POLYGON ((longitude1 latitude1, longitude2 latitude2, ...))**. Note that there is a space after **POLYGON**. Longitudes and latitudes are separated by spaces. Each pair of longitude and latitude is separated by a comma (,). The longitudes and latitudes at the start and end of a polygon must be the same. **IN_POLYGON_LIST** requires at least two polygons. Example: POLYGON ((116.137676 40.163503, 116.137676 39.935276, 116.560993 39.935276, 116.137676 40.163503))
opType	String	Performs union, intersection, and subtraction on multiple polygons. Currently, the following operation types are supported: ● OR: A U B U C (Assume that three polygons A, B, and C are input.) ● AND: A ∩ B ∩ C

UDF output parameter

Parameter	Type	Description
inOrNot	Boolean	Checks whether data is in the specified **polygon_list**.

Example:

```
select longitude, latitude from geosot where IN_POLYGON_LIST('POLYGON ((120.176433 30.327431,120.171283 30.322245,120.181411 30.314540, 120.190509 30.321653,120.185188 30.329358,120.176433 30.327431)), POLYGON ((120.191603 30.328946,120.184179 30.327465,120.181819 30.321464, 120.190359 30.315388,120.199242 30.324464,120.191603 30.328946))', 'OR');
```

- Filtering data based on the polyline list

  IN_POLYLINE_LIST(polylineList, bufferInMeter)

  UDF input parameters

Parameter	Type	Description
polylineList	String	Inputs multiple polylines as a string. Each polyline is presented as **LINESTRING (longitude1 latitude1, longitude2 latitude2, ...)**. Note that there is a space after **LINESTRING**. Longitudes and latitudes are separated by spaces. Each pair of longitude and latitude is separated by a comma (,). A union will be output based on the data in multiple polylines. Example: `LINESTRING (116.137676 40.163503, 116.137676 39.935276, 116.260993 39.935276)`
bufferInMeter	Float	Polyline buffer distance, in meters. Right angles are used at the end to create a buffer.

UDF output parameter

Parameter	Type	Description
inOrNot	Boolean	Checks whether data is in the specified **polyline_list**.

Example:

```
select longitude, latitude from geosot where IN_POLYLINE_LIST('LINESTRING (120.184179 30.327465, 120.191603 30.328946, 120.199242 30.324464, 120.190359 30.315388)', 65);
```

- Filtering data based on the GeoID range list

  IN_POLYGON_RANGE_LIST(polygonRangeList, opType)

  UDF input parameters

Parameter	Type	Description
polygonRangeList	String	Inputs multiple rangeLists as a string. Each rangeList is presented as **RANGELIST (startGeoId1 endGeoId1, startGeoId2 endGeoId2, ...)**. Note that there is a space after **RANGELIST**. Start GeoIDs and end GeoIDs are separated by spaces. Each group of GeoID ranges is separated by a comma (,). Example: RANGELIST (855279368848 855279368850, 855280799610 855280799612, 855282156300 855282157400)
opType	String	Performs union, intersection, and subtraction on multiple rangeLists. Currently, the following operation types are supported: ● OR: A U B U C (Assume that three rangeLists A, B, and C are input.) ● AND: A ∩ B ∩ C

UDF output parameter

Parameter	Type	Description
inOrNot	Boolean	Checks whether data is in the specified **polyRange_list**.

Example:

```
select mygeosot, longitude, latitude from geosot where IN_POLYGON_RANGE_LIST('RANGELIST (526549722865860608 526549722865860618, 532555655580483584 532555655580483594)', 'OR');
```

- Performing polygon query

  IN_POLYGON_JOIN(GEO_HASH_INDEX_COLUMN, POLYGON_COLUMN)

  Perform join query on two tables. One is a spatial data table containing the longitude, latitude, and GeoHashIndex columns, and the other is a dimension table that saves polygon data.

  During query, **IN_POLYGON_JOIN UDF**, **GEO_HASH_INDEX_COLUMN**, and **POLYGON_COLUMN** of the polygon table are used. **Polygon_column** specifies the column containing multiple points (longitude and latitude pairs). The first and last points in each row of the Polygon table must be the same. All points in each row form a closed geometric shape.

  UDF input parameters

Parameter	Type	Description
GEO_HASH_INDEX_COLUMN	Long	GeoHashIndex column of the spatial data table.
POLYGON_COLUMN	String	Polygon column of the polygon table, the value of which is represented by the string of polygon, for example, **POLYGON ((longitude1 latitude1, longitude2 latitude2, ...))**.

Example:

```
CREATE TABLE polygonTable(
polygon string,
poiType string,
poiId String)
STORED AS carbondata;

insert into polygonTable select 'POLYGON ((120.176433 30.327431,120.171283 30.322245,
120.181411 30.314540,120.190509 30.321653,120.185188 30.329358,120.176433 30.327431))','abc','1';

insert into polygonTable select 'POLYGON ((120.191603 30.328946,120.184179 30.327465,
120.181819 30.321464,120.190359 30.315388,120.199242 30.324464,120.191603 30.328946))','abc','2';

select t1.longitude,t1.latitude from geosot t1
inner join
(select polygon,poiId from polygonTable where poitype='abc') t2
on in_polygon_join(t1.mygeosot,t2.polygon) group by t1.longitude,t1.latitude;
```

- Performing range_list query

  IN_POLYGON_JOIN_RANGE_LIST(GEO_HASH_INDEX_COLUMN, POLYGON_COLUMN)

  Use the **IN_POLYGON_JOIN_RANGE_LIST** UDF to associate the spatial data table with the polygon dimension table based on **Polygon_RangeList**. By using a range list, you can skip the conversion between a polygon and a range list.

  UDF input parameters

Parameter	Type	Description
GEO_HASH_INDEX_COLUMN	Long	GeoHashIndex column of the spatial data table.
POLYGON_COLUMN	String	Rangelist column of the Polygon table, the value of which is represented by the string of rangeList, for example, **RANGELIST (startGeoId1 endGeoId1, startGeoId2 endGeoId2, ...)**.

Example:

```
CREATE TABLE polygonTable(
polygon string,
```



```
poiType string,
poiId String)
STORED AS carbondata;

insert into polygonTable select 'RANGELIST (526546455897309184 526546455897309284,
526549831217315840 526549831217315850, 532555655580483534 532555655580483584)','xyz','2';

select t1.*
from geosot t1
inner join
(select polygon,poiId from polygonTable where poitype='xyz') t2
on in_polygon_join_range_list(t1.mygeosot,t2.polygon);
```

UDFs of spacial index tools

- Obtaining row number and column number of a grid converted from GeoID

  GeoIdToGridXy(geoId)

  UDF input parameter

Parameter	Type	Description
geoId	Long	Calculates the row number and column number of the grid based on GeoID.

  UDF output parameter

Parameter	Type	Description
gridArray	Array[Int]	Returns the grid row and column numbers contained in GeoID in array. The first digit indicates the row number, and the second digit indicates the column number.

  Example:

```
select longitude, latitude, mygeohash, GeoIdToGridXy(mygeohash) as GridXY from geoTable;
```

- Converting longitude and latitude to GeoID

  LatLngToGeoId(latitude, longitude oriLatitude, gridSize)

  UDF input parameters

Parameter	Type	Description
longitude	Long	Longitude. Note: The value is an integer after conversion.
latitude	Long	Latitude. Note: The value is an integer after conversion.
oriLatitude	Double	Origin latitude, required for calculating GeoID.

Parameter	Type	Description
gridSize	Int	Grid size, required for calculating GeoID.

UDF output parameter

Parameter	Type	Description
geoId	Long	Returns a number that indicates the longitude and latitude after coding.

Example:

```
select longitude, latitude, mygeohash, LatLngToGeoId(latitude, longitude, 39.832277, 50) as geoId
from geoTable;
```

- Converting GeoID to longitude and latitude

  GeoIdToLatLng(geoId, oriLatitude, gridSize)

  UDF input parameters

Parameter	Type	Description
geoId	Long	Calculates the longitude and latitude based on GeoID.
oriLatitude	Double	Origin latitude, required for calculating the longitude and latitude.
gridSize	Int	Grid size, required for calculating the longitude and latitude.

  NOTE

  GeoID is generated based on the grid coordinates, which are the grid center. Therefore, the calculated longitude and latitude are the longitude and latitude of the grid center. There may be an error ranging from 0 degree to half of the grid size between the calculated longitude and latitude and the longitude and latitude of the generated GeoID.

  UDF output parameter

Parameter	Type	Description
latitudeAndLongitude	Array[Double]	Returns the longitude and latitude coordinates of the grid center that represent the GeoID in array. The first digit indicates the latitude, and the second digit indicates the longitude.

Example:

```
select longitude, latitude, mygeohash, GeoIdToLatLng(mygeohash, 39.832277, 50) as
LatitudeAndLongitude from geoTable;
```

- Calculating the upper-layer GeoID of the pyramid model

  ToUpperLayerGeoId(geoId)

  UDF input parameter

Parameter	Type	Description
geoId	Long	Calculates the upper-layer GeoID of the pyramid model based on the input GeoID.

  UDF output parameter

Parameter	Type	Description
geoId	Long	Returns the upper-layer GeoID of the pyramid model.

  Example:

```
select longitude, latitude, mygeohash, ToUpperLayerGeoId(mygeohash) as upperLayerGeoId from
geoTable;
```

- Obtaining the GeoID range list using the input polygon

  ToRangeList(polygon, oriLatitude, gridSize)

  UDF input parameters

Parameter	Type	Description
polygon	String	Input polygon string, which is a pair of longitude and latitude. Longitude and latitude are separated by a space. Each pair of longitude and latitude is separated by a comma (,). The longitude and latitude at the start and end must be the same.
oriLatitude	Double	Origin latitude, required for calculating GeoID.
gridSize	Int	Grid size, required for calculating GeoID.

  UDF output parameter

Parameter	Type	Description
geoIdList	Buffer[Array[Long]]	Converts polygons into GeoID range lists.

Example:

```
select ToRangeList('116.321011 40.123503, 116.137676 39.947911, 116.560993 39.935276, 116.321011 40.123503', 39.832277, 50) as rangeList from geoTable;
```

- Calculating the upper-layer longitude of the pyramid model

  ToUpperLongitude (longitude, gridSize, oriLat)

  UDF input parameters

Parameter	Type	Description
longitude	Long	Input longitude, which is a long integer.
gridSize	Int	Grid size, required for calculating longitude.
oriLatitude	Double	Origin latitude, required for calculating longitude.

  UDF output parameter

Parameter	Type	Description
longitude	Long	Returns the upper-layer longitude.

  Example:

```
select ToUpperLongitude (-23575161504L, 50, 39.832277) as upperLongitude from geoTable;
```

- Calculating the upper-layer latitude of the pyramid model

  ToUpperLatitude(Latitude, gridSize, oriLat)

  UDF input parameters

Parameter	Type	Description
latitude	Long	Input latitude, which is a long integer.
gridSize	Int	Grid size, required for calculating latitude.
oriLatitude	Double	Origin latitude, required for calculating latitude.

UDF output parameter

Parameter	Type	Description
Latitude	Long	Returns the upper-layer latitude.

Example:

```
select ToUpperLatitude (-23575161504L, 50, 39.832277) as upperLatitude from geoTable;
```

- Converting longitude and latitude to GeoSOT

  LatLngToGridCode(latitude, longitude, level)

  UDF input parameters

Parameter	Type	Description
latitude	Double	Latitude.
longitude	Double	Longitude.
level	Int	Level. The value range is [0, 32].

UDF output parameter

Parameter	Type	Description
geoId	Long	A number that indicates the longitude and latitude after GeoSOT encoding.

Example:

```
select LatLngToGridCode(39.930753, 116.302895, 21) as geoId;
```

1.8 Common Issues About CarbonData

1.8.1 Why Is Incorrect Output Displayed When I Perform Query with Filter on Decimal Data Type Values?

Question

Why is incorrect output displayed when I perform query with filter on decimal data type values?

For example:

select * from carbon_table where num = 1234567890123456.22;

Output:

```
+------+----------------------+--+
| name |        num           |
+------+----------------------+--+
| IAA  | 1234567890123456.22 |
| IAA  | 1234567890123456.21 |
+------+----------------------+--+
```

Answer

To obtain accurate output, append BD to the number.

For example:

select * from carbon_table where num = 1234567890123456.22BD;

Output:

```
+------+----------------------+--+
| name |        num           |
+------+----------------------+--+
| IAA  | 1234567890123456.22 |
+------+----------------------+--+
```

1.8.2 How to Avoid Minor Compaction for Historical Data?

Question

How to avoid minor compaction for historical data?

Answer

If you want to load historical data first and then the incremental data, perform following steps to avoid minor compaction of historical data:

1. Load all historical data.
2. Configure the major compaction size to a value smaller than the segment size of historical data.
3. Run the major compaction once on historical data so that these segments will not be considered later for minor compaction.
4. Load the incremental data.
5. You can configure the minor compaction threshold as required.

For example:

1. Assume that you have loaded all historical data to CarbonData and the size of each segment is 500 GB.
2. Set the threshold of major compaction property to **carbon.major.compaction.size** = **491520** (480 GB x 1024).
3. Run major compaction. All segments will be compacted because the size of each segment is more than configured size.
4. Perform incremental loading.
5. Configure the minor compaction threshold to **carbon.compaction.level.threshold** = **6,6**.
6. Run minor compaction. As a result, only incremental data is compacted.

1.8.3 How to Change the Default Group Name for CarbonData Data Loading?

Question

How to change the default group name for CarbonData data loading?

Answer

By default, the group name for CarbonData data loading is **ficommon**. You can perform the following operation to change the default group name:

1. Edit the **carbon.properties** file.
2. Change the value of the key **carbon.dataload.group.name** as required. The default value is **ficommon**.

1.8.4 Why Does INSERT INTO CARBON TABLE Command Fail?

Question

Why does the ***INSERT INTO CARBON TABLE*** command fail and the following error message is displayed?

```
Data load failed due to bad record
```

Answer

The ***INSERT INTO CARBON TABLE*** command fails in the following scenarios:

- If the data type of source and target table columns are not the same, the data from the source table will be treated as bad records and the ***INSERT INTO*** command fails.
- If the result of aggregation function on a source column exceeds the maximum range of the target column, then the ***INSERT INTO*** command fails.

  Solution:

  You can use the cast function on corresponding columns when inserting records.

  For example:

  a. Run the ***DESCRIBE*** command to query the target and source table.

     DESCRIBE *newcarbontable*;

     Result:

     ```
     col1 int
     col2 bigint
     ```

     DESCRIBE *sourcetable*;

     Result:

     ```
     col1 int
     col2 int
     ```

  b. Add the cast function to convert bigint value to integer.

     INSERT INTO *newcarbontable select col1, cast(col2 as integer) from sourcetable;*

1.8.5 Why Is the Data Logged in Bad Records Different from the Original Input Data with Escape Characters?

Question

Why is the data logged in bad records different from the original input data with escaped characters?

Answer

An escape character is a backslash (\) followed by one or more characters. If the input records contain escape characters such as \t, \b, \n, \r, \f, \', \", \\ , java will process the escape character '\' and the following characters together to obtain the escaped meaning.

For example, if the CSV data type **2010\\10,test** is inserted to String,int type, the value is treated as bad records, because **tes**t cannot be converted to int. The value logged in the bad records is **2010\10** because java processes **** as ****.

1.8.6 Why Data Load Performance Decreases due to Bad Records?

Question

Why data load performance decreases due to bad records?

Answer

The loaded data contains Bad Records, and the value of **BAD_RECORDS_LOGGER_ENABLE** is **true** or **BAD_RECORDS_ACTION** is **redirect**.

Data load performance degrades due to additional I/O overhead caused by writing failure causes to the log file or redirecting Bad Records to the original CSV file.

1.8.7 Why Data loading Fails During off heap?

Question

Why Data Loading fails during off heap?

Answer

YARN Resource Manager will consider (Java heap memory + **spark.yarn.am.memoryOverhead**) as memory limit, so during the off heap, the memory can exceed this limit.

You need to set **spark.yarn.am.memoryOverhead** in the **spark-defaults.conf** file to increase the memory size.

1.8.8 Why Do I Fail to Create a Hive Table?

Question

Why do I fail to create a hive table?

Answer

Creating a Hive table fails, when source table or sub query has more number of partitions. The implementation of the query requires a lot of tasks, then the number of files will be output a lot, resulting OOM in Driver.

It can be solved by using ***distribute by*** on suitable cardinality(distinct values) column in the statement of Hive table creation.

distribute by clause limits number of hive table partitions. It considers cardinality of given column or **spark.sql.shuffle.partitions** which ever is minimal. For example, if **spark.sql.shuffle.partitions** is 200, but cardinality of column is 100, out files is 200, but the other 100 files are empty. So using very low cardinality column like 1 will cause data skew and will effect later query distribution.

So we suggest using the column with cardinality greater than **spark.sql.shuffle.partitions**. It can be greater than 2 to 3 times.

Example:

create table hivetable1 as select * from sourcetable1 distribute by col_age;

1.8.9 How Do I Logically Split Data Across Different Namespaces?

Question

How do I logically split data across different namespaces?

Answer

- Configuration:

  To logically split data across different namespaces, you must update the following configuration in the **core-site.xml** file of HDFS, Hive, and Spark.

  NOTE

  Changing the Hive component will change the locations of carbonstore and warehouse.

  - Configuration in HDFS
    - **fs.defaultFS**: Name of the default file system. The URI mode must be set to **viewfs**. When **viewfs** is used, the permission part must be **ClusterX**.
    - **fs.viewfs.mountable.ClusterX.homedir**: Home directory base path. You can use the getHomeDirectory() method defined in **FileSystem/FileContext** to access the home directory.

- fs.viewfs.mountable.default.link.<dir_name>: ViewFS mount table.

Example:

```
<property>
<name>fs.defaultFS</name>
<value>viewfs://ClusterX/</value>
</property>
<property>
<name>fs.viewfs.mounttable.ClusterX.link./folder1</name>
<value>hdfs://NS1/folder1</value>
</property>
<property>
<name>fs.viewfs.mounttable.ClusterX.link./folder2</name>
<value>hdfs://NS2/folder2</value>
</property>
```

– Configurations in Hive and Spark

fs.defaultFS: Name of the default file system. The URI mode must be set to **viewfs**. When **viewfs** is used, the permission part must be **ClusterX**.

- Syntax:

  LOAD DATA INPATH *'path to data' INTO TABLE table_name OPTIONS ('...');*

  NOTE

  When Spark is configured with the viewFS file system and attempts to load data from HDFS, users must specify a path such as **viewfs://** or a relative path as the file path in the **LOAD** statement.

- Example:
  – Sample viewFS path:

    LOAD DATA INPATH *'viewfs://ClusterX/dir/data.csv' INTO TABLE table_name OPTIONS ('...');*

  – Sample relative path:

    LOAD DATA INPATH *'/apps/input_data1.txt'* ***INTO TABLE*** *table_name*;

1.8.10 Why the UPDATE Command Cannot Be Executed in Spark Shell?

Question

Why the UPDATE command cannot be executed in Spark Shell?

Answer

The syntax and examples provided in this document are about Beeline commands instead of Spark Shell commands.

To run the UPDATE command in Spark Shell, use the following syntax:

- Syntax 1

  <carbon_context>.sql("UPDATE <CARBON TABLE> SET (column_name1, column_name2, ... column_name n) = (column1_expression , column2_expression , column3_expression ... column n_expression) [WHERE { <filter_condition> }];").show

- Syntax 2

<carbon_context>.sql("UPDATE <CARBON TABLE> SET (column_name1, column_name2,) = (select sourceColumn1, sourceColumn2 from sourceTable [WHERE { <filter_condition> }]) [WHERE { <filter_condition> }];").show

Example:

If the context of CarbonData is **carbon**, run the following command:

carbon.sql("update carbonTable1 d set (d.column3,d.column5) = (select s.c33 ,s.c55 from sourceTable1 s where d.column1 = s.c11) where d.column1 = 'country' exists(select * from table3 o where o.c2 > 1);").show

1.8.11 How Do I Configure Unsafe Memory in CarbonData?

Question

How do I configure unsafe memory in CarbonData?

Answer

In the Spark configuration, the value of **spark.yarn.executor.memoryOverhead** must be greater than the sum of (**sort.inmemory.size.inmb** + **Netty offheapmemory required**), or the sum of (**carbon.unsafe.working.memory.in.mb** + **carbon.sort.inememory.storage.size.in.mb** + **Netty offheapmemory required**). Otherwise, if off-heap access exceeds the configured executor memory, Yarn may stop the executor.

If **spark.shuffle.io.preferDirectBufs** is set to **true**, the netty transfer service in Spark takes some heap memory (around 384 MB or 0.1 x executor memory) from **spark.yarn.executor.memoryOverhead**.

For details, see **Configuring Heap Memory Parameters for Spark Executor**.

1.8.12 Why Does CarbonData Become Abnormal After the Disk Space Quota of the HDFS Storage Directory Is Set?

Question

Why does CarbonData become abnormal after the disk space quota of the HDFS storage directory is set?

Answer

When a table is created, loaded, or updated, or other operations are performed, data is written to HDFS. If the disk space quota of the HDFS directory is insufficient, the operation fails and the exception is thrown.

```
org.apache.hadoop.hdfs.protocol.DSQuotaExceededException: The DiskSpace quota of /user/tenant is exceeded: quota = 314572800 B = 300 MB but diskspace consumed = 402653184 B = 384 MB at org.apache.hadoop.hdfs.server.namenode.DirectoryWithQuotaFeature.verifyStoragespaceQuota(DirectoryWithQuotaFeature.java:211) at org.apache.hadoop.hdfs.server.namenode.DirectoryWithQuotaFeature.verifyQuota(DirectoryWithQuotaFeature.java:239) at org.apache.hadoop.hdfs.server.namenode.FSDirectory.verifyQuota(FSDirectory.java:941) at org.apache.hadoop.hdfs.server.namenode.FSDirectory.updateCount(FSDirectory.java:745)
```


You need to set more disk space quota for the tenant.

The following is an example:

The required disk space can be calculated as follows:

If the number of HDFS replicas is 3 and the default block size is 128 MB, at least 384 MB disk space is required in the HDFS for writing table schema files. Formula: Number of block x **block_size** x **replication_factor** of the schema file = 1 x 128 x 3 = 384 MB

NOTE

When you load data, reserve at least 3072 MB for each **fact** file as the default block size is 1024 MB.

1.8.13 Why Do Files of a Carbon Table Exist in the Recycle Bin Even If the drop table Command Is Not Executed When Mis-deletion Prevention Is Enabled?

Question

Why do files of a Carbon table exist in the recycle bin even if the **drop table** command is not executed when mis-deletion prevention is enabled?

Answer

After the mis-deletion prevention is enabled for a Carbon table, calling a file deletion command will move the deleted files to the recycle bin.

The intermediate file **.carbonindex** is deleted duration the execution of the **insert** or **load** command. Therefore, the table files may exist in the recycle bin even through the **drop table** command is not executed.

If you run the **drop table** command, a table directory with a timestamp is generated. The files in the directory are complete.

1.8.14 How Do I Restore the Latest tablestatus File That Has Been Lost or Damaged When TableStatus Versioning Is Enabled?

Question

When the TableStatus versioning feature is enabled, how do I restore the latest **tablestatus** file if it is lost or damaged due to other exceptions?

Answer

Use the latest available **tablestatus** file to restore data in the following scenarios:

Scenario 1: The CarbonData data files and .segment files of the current batch are damaged and cannot be restored.

1. Log in to the client node and run the following commands to view the **tablestatus** file of the HDFS table and find the latest tablestatus version number:

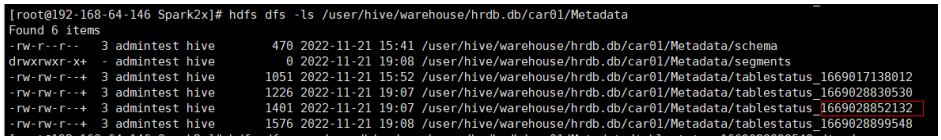
   cd *Client installation path*

   source bigdata_env

   source Spark/component_env

   kinit *Component service user* (You do not need to run the **kinit** command for normal clusters.)

   hdfs dfs -ls /user/hive/warehouse/hrdb.db/car01/Metadata

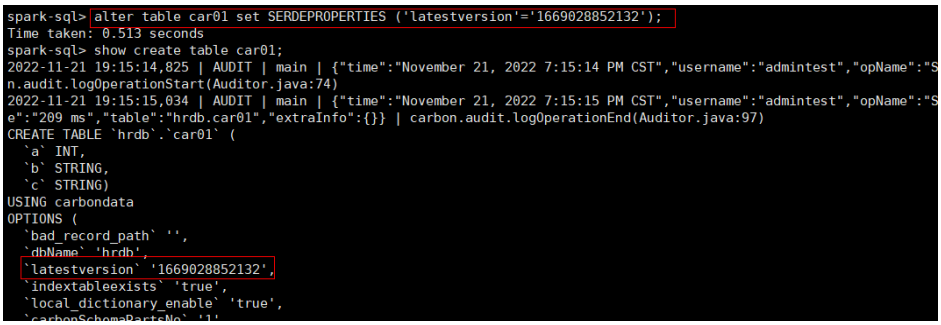


   NOTE

   In the preceding figure, the **tablestatus_1669028899548** file of the current batch is damaged and the **tablestatus_1669028852132** file is required.

2. Go to Spark SQL and run the following command to change the value of **latestversion** to the latest version:

   alter table car01 set SERDEPROPERTIES ('latestversion'='1669082252132');



   You need to exit the current session, reconnect to the session, and perform the query. This method has been used to restore customer data as much as possible. Generally, segment data files on the live network cannot be restored in power-off scenarios.

Scenario 2: The CarbonData data files and .segment files of the current batch are complete and can be restored.

Use the TableStatusRecovery tool to restore non-partitioned tables. Log in to the Spark client node and run the following commands:

cd *Client installation path*

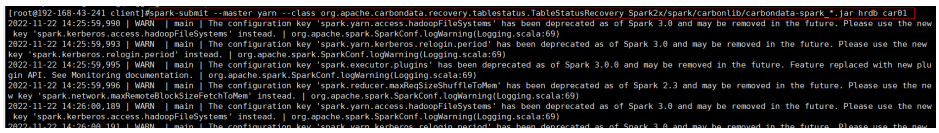
source bigdata_env

source Spark/component_env

kinit *Component service user* (You do not need to run the **kinit** command for normal clusters.)

spark-submit --master yarn --class org.apache.carbondata.recovery.tablestatus.TableStatusRecovery Spark/spark/carbonlib/carbondata-spark_*.jar *hrdb car01*

Parameter description: **hrdb car01** indicates the table name.



Restrictions on using TableStatusRecovery for restoration:

- After the merge, if the **tablestatus** file is lost or damaged, this tool cannot be used to restore the segments in the merge state because only the **tablestatus** file contains the segment merge information.
- After segments are deleted by ID or date, if the **tablestatus** file is lost or damaged, the deleted segment information cannot be restored because only the **tablestatus** file contains the segment deletion information.
- This tool cannot be used on materialized view tables.
- If the latest **tablestatus** file is faulty and query cannot be performed after using this tool for restoration, remove this latest file and use the previous **tablestatus** file for restoration.

1.9 CarbonData Troubleshooting

1.9.1 Filter Result Is not Consistent with Hive when a Big Double Type Value Is Used in Filter

Symptom

When double data type values with higher precision are used in filters, incorrect values are returned by filtering results.

Possible Causes

When double data type values with higher precision are used in filters, values are rounded off before comparison. Therefore, values of double data type with different fraction part are considered same.

Troubleshooting Method

NA.

Procedure

To avoid this problem, use decimal data type when high precision data comparisons are required, such as financial applications, equality and inequality checks, and rounding operations.

Reference Information

NA.

1.9.2 Query Performance Deteriorated Due to Insufficient Executor Memory

Symptom

The query performance fluctuates when the query is executed in different query periods.

Possible Causes

During data loading, the memory configured for each executor program instance may be insufficient, resulting in more Java GCs. When GC occurs, the query performance deteriorates.

Troubleshooting Method

On the Spark UI, the GC time of some executors is obviously higher than that of other executors, or all executors have high GC time.

Procedure

Log in to Manager and choose **Cluster** > **Services** > **Spark2x**. On the displayed page, click the **Configurations** tab and then **All Configurations**, search for **spark.executor.memory** in the search box, and set its value to a larger value.

Reference

None

1.9.3 Data Query or Loading Failed, and "org.apache.carbondata.core.memory.MemoryException: Not enough memory" Was Reported

Question

Why does data query or loading fail and "org.apache.carbondata.core.memory.MemoryException: Not enough memory" is displayed?

Answer

This exception is thrown when the out-of-heap memory required for data query and loading in the executor is insufficient.

In this case, increase the values of **carbon.unsafe.working.memory.in.mb** and **spark.yarn.executor.memoryOverhead**.

For details, see **How Do I Configure Unsafe Memory in CarbonData?**.

The memory is shared by data query and loading. Therefore, if the loading and query operations need to be performed at the same time, you are advised to set **carbon.unsafe.working.memory.in.mb** and **spark.yarn.executor.memoryOverhead** to a value greater than 2,048 MB.

The following formula can be used for estimation:

Memory required for data loading:

carbon.number.of.cores.while.loading [default value is 6] x Number of tables to load in parallel x offheap.sort.chunk.size.inmb [default value is 64 MB] + carbon.blockletgroup.size.in.mb [default value is 64 MB] + Current compaction ratio [64 MB/3.5])

= Around 900 MB per table

Memory required for data query:

(SPARK_EXECUTOR_INSTANCES. [default value is 2] x (carbon.blockletgroup.size.in.mb [default value: 64 MB] + carbon.blockletgroup.size.in.mb [default value = 64 MB x 3.5) x Number of cores per executor [default value: 1])

= ~ 600 MB

1.9.4 Why INSERT INTO/LOAD DATA Task Distribution Is Incorrect and the Opened Tasks Are Less Than the Available Executors when the Number of Initial Executors Is Zero?

Question

Why **INSERT INTO or LOAD DATA** task distribution is incorrect, and the openedtasks are less than the available executors when the number of initial executors is zero**?**

Answer

In case of INSERT INTO or LOAD DATA, CarbonData distributes one task per node. If the executors are not allocated from the distinct nodes then CarbonData will launch fewer tasks.

Solution:

Configure higher value for the executor memory and core so that the yarn can launch only one executor per node.

1. Configure the number of the Executor cores.
   - Configure the **spark.executor.cores** in **spark-defaults.conf** or the **SPARK_EXECUTOR_CORES** in **spark-env.sh** appropriately.
   - Add **--executor-cores NUM** parameter to configure the cores during use the spark-submit command.

2. Configure the Executor memory.
   - Configure the **spark.executor.memory** in **spark-defaults.conf** or the **SPARK_EXECUTOR_MEMORY** in **spark-env.sh** appropriately.
   - Add **--executor-memory MEM** parameter to configure the memory during use the spark-submit command.

1.9.5 Why Does CarbonData Require Additional Executors Even Though the Parallelism Is Greater Than the Number of Blocks to Be Processed?

Question

Why does CarbonData require additional executors even though the parallelism is greater than the number of blocks to be processed?

Answer

CarbonData block distribution optimizes data processing as follows:

1. Optimize data processing parallelism.
2. Optimize parallel reading of block data.

To optimize parallel processing and parallel read, CarbonData requests executors based on the locality of blocks so that it can obtain executors on all nodes.

If you are using dynamic allocation, you need to configure the following properties:

1. Set **spark.dynamicAllocation.executorIdleTimeout** to 15 minutes (or the average query time).
2. Set **spark.dynamicAllocation.maxExecutors** correctly. The default value **2048** is not recommended. Otherwise, CarbonData will request the maximum number of executors.
3. For a bigger cluster, set **carbon.dynamicAllocation.schedulerTimeout** to a value ranging from 10 to 15 seconds. The default value is 5 seconds.
4. Set **carbon.scheduler.minRegisteredResourcesRatio** to a value ranging from 0.1 to 1.0. The default value is **0.8**. Block distribution can be started as long as the value of **carbon.scheduler.minRegisteredResourcesRatio** is within the range.

2 Using CDL

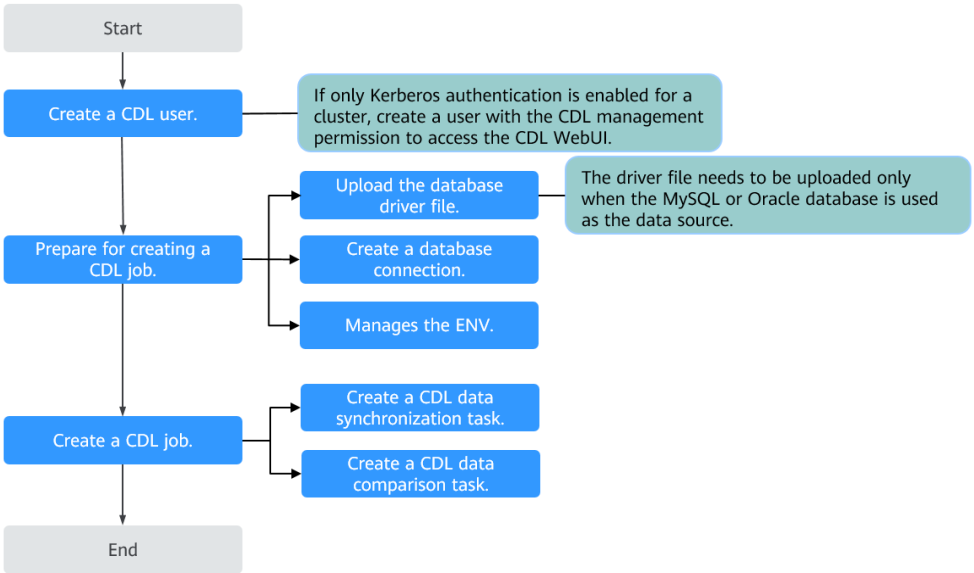
2.1 Integrating CDL Data

CDL is a simple and efficient real-time data integration service. It captures data change events from various OLTP databases and pushes them to Kafka. The Sink Connector consumes data in topics and imports the data to the software applications of big data ecosystems. In this way, data is imported to the data lake in real time.

The CDL service contains two roles: CDLConnector and CDLService. CDLConnector is the instance for executing a data capture job, and CDLService is the instance for managing and creating a job.

You can create data synchronization and comparison tasks on the CDLService WebUI. Figure 2-1 shows the process of using the CDL.

Figure 2-1 CDL usage process



Data synchronization task

- The CDL supports the following types of data synchronization tasks:

  Table 2-1 Data synchronization task types supported by the CDL

Data source	Destination end	Description
MySQL	Hudi	This task synchronizes data from the MySQL database to Hudi.
	Kafka	This task synchronizes data from the MySQL database to Kafka.
PgSQL	Hudi	This task synchronizes data from the PgSQL database to Hudi.
	Kafka	This task synchronizes data from the PgSQL database to Kafka.
Hudi	DWS	This task synchronizes data from the Hudi database to DWS.
	ClickHouse	This task synchronizes data from the Hudi database to ClickHouse.
ThirdKafka	Hudi	This task synchronizes data from the ThirdKafka database to Hudi.
	Kafka	This task synchronizes data from the ThirdKafka database to Kafka.
openGauss (supported by MRS 3.3.0 and later versions)	ThirdKafka (DMS/DRS) -> Hudi	Synchronizes data from GaussDB(for openGauss) to Hudi through ThirdKafka (DMS/DRS).
	Hudi	Synchronizes data from openGauss to Hudi.
	Kafka	Synchronizes data from openGauss to Kafka.
ogg-oracle-avro (supported by MRS 3.3.0 and later versions)	ThirdKafka (DMS/DRS) -> Hudi	Synchronizes data from avro-oracle to Hudi through ThirdKafka (DMS/DRS).

- Database versions supported by CDL in a data synchronization task:

  Kafka (including ThirdKafka that uses MRS Kafka as the source), Hudi, and ClickHouse use related components in the MRS cluster as data sources. For

details about the version numbers, see **List of MRS Component Versions**. **Table 2-2** lists the database versions.

Table 2-2 Database versions

Database Name	Data Source	Destination End	Version
MySQL	Support	Not supported	5.7 and 8.0.*x*
PostgreSQL	Support	Not supported	9.6, 10, 11, 12, and 13
Opengauss (Open source version)	Support	Not supported	2.1.0 or later
DWS	Not supported	Support	8.1.1 or later

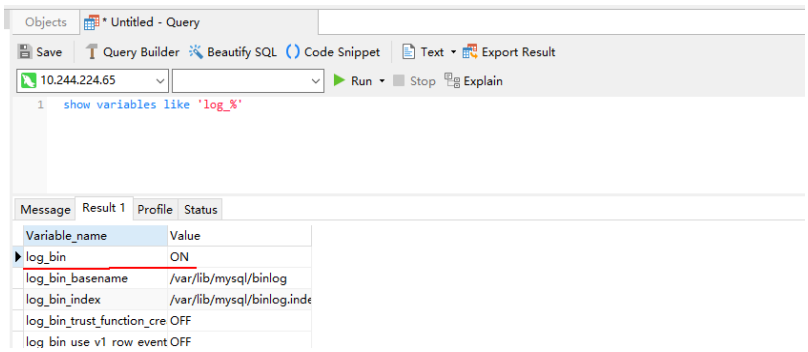
- Usage Constraints:
  - If CDL is required, the value of **log.cleanup.policy** of Kafka must be **delete**.
  - The CDL service has been installed in the MRS cluster.
  - CDL can capture incremental data only from non-system tables, but not from built-in databases of databases such as MySQL, and PostgreSQL.
  - When data is synchronized from Hudi to DWS or ClickHouse, the data that is deleted from Hudi will not be deleted from the destination. For example, if you run the **delete from** *tableName* command in Hudi to physically delete table data, the table data still exists on the destination DWS or ClickHouse.
  - Binary logging (enabled by default) and GTID have been enabled for the MySQL database. CDL cannot fetch tables whose names contain special characters such as the dollar sign ($) or Chinese characters.

    To check whether binary logging is enabled for the MySQL database:

    Use a tool (Navicat is used in this example) or CLI to connect to the MySQL database and run the **show variables like 'log_%'** command to view the configuration.

    For example, in Navicat, choose **File** > **New Query** to create a query, enter the following SQL statement, and click **Run**. If **log_bin** is displayed as **ON** in the result, the function is enabled successfully.

    show variables like 'log_%'

If the bin log function of the MySQL database is not enabled, perform the following operations:

Modify the MySQL configuration file **my.cnf** (**my.ini** for Windows) as follows:

```
server-id        = 223344
log_bin          = mysql-bin
binlog_format    = ROW
binlog_row_image  = FULL
expire_logs_days  = 10
```

After the modification, restart MySQL for the configurations to take effect.

To check whether GTID is enabled for the MySQL database:

Run the **show global variables like '%gtid%'** command to check whether GTID is enabled. For details, see the official documentation of the corresponding MySQL version. (For details about how to enable the function in MySQL 8.x, see **https://dev.mysql.com/doc/refman/8.0/en/replication-mode-change-online-enable-gtids.html**.)

```
mysql> show global variables like '%gtid%';
+---------------------------------+-------------------------------------------+
| Variable_name                   | Value                                     |
+---------------------------------+-------------------------------------------+
| binlog_gtid_simple_recovery     | ON                                        |
| enforce_gtid_consistency        | ON                                        |
| gtid_executed                   | da708678-09f9-11eb-8d03-fa163e62b70b:1-41 |
| gtid_executed_compression_period | 1000                                      |
| gtid_mode                       | ON                                        |
| gtid_owned                      |                                           |
| gtid_purged                     |                                           |
| session_track_gtids             | OFF                                       |
+---------------------------------+-------------------------------------------+
8 rows in set (0.07 sec)
```

Set user permissions:

To execute MySQL tasks, users must have the **SELECT**, **RELOAD**, **SHOW DATABASES**, **REPLICATION SLAVE** and **REPLICATION CLIENT** permissions.

Run the following command to grant the permissions, Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

GRANT SELECT, RELOAD, SHOW DATABASES, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO '*Username*' **IDENTIFIED BY** '*Password*';

Run the following command to update the permissions:

FLUSH PRIVILEGES;

- The write-ahead log policy is modified for the PostgreSQL database.

NOTE

- The user for connecting to the PostgreSQL database must have the replication permission, the CREATE permission on the database, and is the owner of tables.
- CDL cannot fetch tables whose names contain special characters such as the dollar sign ($) or Chinese characters.
- For PostgreSQL databases, you must have the permission to set the **statement_timeout** and **lock_timeout** parameters and the permission to query and delete slots and publications.
- You are advised to set **max_wal_senders** to 1.5 or 2 times the value of **Slot**.
- If the replication identifier of a PostgreSQL table is **default**, enable the full field completion function in the following scenarios:
  - Scenario 1:

    When the **delete** operation is performed on the source database, a **delete** event contains only the primary key information. In this case, for the **delete** data written to Hudi, only the primary key has values, and the values of other service fields are **null**.
  - Scenario 2:

    When the size of a single piece of data in the database exceeds 8 KB (including 8 KB), an **update** event contains only changed fields. In this case, the values of some fields in the Hudi data are **__debezium_unavailable_value**.

  The related commands are as follows:
  - Command for querying the replication identifier of a PostgreSQL table:

    SELECT CASE relreplident WHEN 'd' THEN 'default' WHEN 'n' THEN 'nothing' WHEN 'f' THEN 'full' WHEN 'i' THEN 'index' END AS replica_identity FROM pg_class WHERE oid = '*tablename***'::regclass;**
  - Command for enabling the full field completion function for a table:

    ALTER TABLE *tablename* **REPLICA IDENTITY FULL;**

i. Modify **wal_level = logical** in the database configuration file **postgresql.conf** (which is stored in the **data** folder in the PostgreSQL installation directory by default).

```
#------------------------------------------------
#WRITE-AHEAD LOG
#------------------------------------------------

# - Settings -
wal_level = logical         # minimal, replica, or logical
                    # (change requires restart)
#fsync = on             #flush data to disk for crash safety
...
```

ii. Restart the database service.

```
# Stop
pg_ctl stop
# Start
pg_ctl start
```

– Prerequisites for the DWS database

Before a synchronization task is started, both the source and target tables exist and have the same table structure. The value of **ads_last_update_date** in the DWS table is the current system time.

– Prerequisites for ThirdPartyKafka

The upper-layer source supports openGauss and OGG. Kafka topics at the source end can be consumed by Kafka in the MRS cluster.

NOTE

ThirdKafka does not support data synchronization from the distributed openGauss database to CDL.

- Prerequisites for ClickHouse

  You have the permissions to operate ClickHouse. For details, see **Creating a ClickHouse Role**.

- Write-ahead logging required for the openGauss database (supported in MRS 3.3.0 and later versions)

  NOTE

  - The user connecting to openGauss databases must have logical replication permission.
  - openGauss interval partitioned tables and Chinese table names cannot be synchronized.
  - If a table in the source openGauss database does not have a primary key, data in the table cannot be deleted.
  - If the data in the source openGauss database needs to be deleted, run the following commands to enable the field completion function:
    - Command for querying the replication identifier of a openGauss table:

      SELECT CASE relreplident WHEN 'd' THEN 'default' WHEN 'n' THEN 'nothing' WHEN 'f' THEN 'full' WHEN 'i' THEN 'index' END AS replica_identity FROM pg_class WHERE oid = '*tablename***'::regclass;**
    - Command for enabling the full field completion function for a table:

      ALTER TABLE *tablename* **REPLICA IDENTITY FULL;**

  i. Modify **wal_level = logical** in the database configuration file **postgresql.conf** (which is stored in the **data** folder in the openGauss installation directory by default).

  ```
  #------------------------------------------------
  #WRITE-AHEAD LOG
  #------------------------------------------------

  # - Settings -
  wal_level = logical         # minimal, replica, or logical
                              # (change requires restart)
  #fsync = on                 #flush data to disk for crash safety
  ...
  ```

  ii. Restart the database service.

  ```
  # Stop
  pg_ctl stop
  # Start
  pg_ctl start
  ```

Data Types and Mapping Supported by CDL Synchronization Tasks

This section describes the data types supported by CDL synchronization tasks and the mapping between data types of the source database and Spark data types.

Table 2-3 Mapping between PostgreSQL and Spark data types

PostgreSQL Data Type	Spark (Hudi) Data Type
int2	int

PostgreSQL Data Type	Spark (Hudi) Data Type
int4	int
int8	bigint
numeric(p, s)	decimal[p,s]
bool	boolean
char	string
varchar	string
text	string
timestamptz	timestamp
timestamp	timestamp
date	date
json, jsonb	string
float4	float
float8	double

Table 2-4 Mapping between MySQL and Spark data types

MySQL Data Type	Spark (Hudi) Data Type
int	int
integer	int
bigint	bigint
double	double
decimal[p,s]	decimal[p,s]
varchar	string
char	string
text	string
timestamp	timestamp
datetime	timestamp
date	date
json	string
float	double

Table 2-5 Mapping between Ogg/Ogg Oracle Avro (MRS 3.3.0 and later versions) and Spark data types

Oracle Data Type	Spark (Hudi) Data Type
NUMBER(3), NUMBER(5)	bigint
INTEGER	decimal
NUMBER(20)	decimal
NUMBER	decimal
BINARY_DOUBLE	double
CHAR	string
VARCHAR	string
TIMESTAMP, DATETIME	timestamp
timestamp with time zone	timestamp
DATE	timestamp

Table 2-6 Mapping Between DRS openGauss JSON types and Spark data types (supported in MRS 3.3.0 and later versions)

openGauss JSON	Spark (Hudi)
int2	int
int4	int
int8	bigint
numeric(p,s)	decimal[p,s]
bool	boolean
varchar	string
timestamp	timestamp
timestampz	timestamp
date	date
jsonb	string
json	string
float4	float
float8	double
text	string

Table 2-7 Mapping between DRS Oracle JSON types and Spark data types (supported in MRS 3.3.0 and later versions)

Oracle JSON	Spark (Hudi)
number(p,s)	decimal[p,s]
binary double	double
char	string
varchar2	string
nvarchar2	string
timestamp	timestamp
timestamp with time zone	timestamp
date	timestamp

Table 2-8 Mapping between DRS Oracle Avro types and Spark data types (supported in MRS 3.3.0 and later versions)

Oracle Avro	Spark (Hudi)
nuber[p,s]	decimal[p,s]
float(p)	float
binary_double	double
char(p)	string
varchar2(p)	string
timestamp(p)	timestamp
date	timestamp

Table 2-9 Mapping between openGauss data types and Spark data types (supported in MRS 3.3.0 and later versions)

openGauss	Spark (Hudi)
int1	int
int2	int
int4	int
int8	bigint
numeric(p,s)	decimal[p,s]

openGauss	Spark (Hudi)
bool	boolean
char	string
bpchar	string
nvarchar2	string
text	string
date	date
timestamp	timestamp
timestampz	timestamp
json	string
jsonb	string
float4	float
float8	double
real	float

Table 2-10 Mapping between Spark (Hudi) and DWS data types

Spark (Hudi) Data Type	DWS Data Type
int	int
long	bigint
float	float
double	double
decimal[p,s]	decimal[p,s]
boolean	boolean
string	varchar
date	date
timestamp	timestamp

Table 2-11 Mapping between Spark (Hudi) and ClickHouse data types

Spark (Hudi) Data Type	ClickHouse Data Type
int	Int32

Spark (Hudi) Data Type	ClickHouse Data Type
long	Int64 (bigint)
float	Float32 (float)
double	Float64 (double)
decimal[p,s]	Decimal(P,S)
boolean	bool
string	String (LONGTEXT, MEDIUMTEXT, TINYTEXT, TEXT, LONGBLOB, MEDIUMBLOB, TINYBLOB, BLOB, VARCHAR, CHAR)
date	Date
timestamp	DateTime

Data comparison task

Data comparison checks the consistency between data in the source database and that in the target Hive. If the data is inconsistent, CDL can attempt to repair the inconsistent data. For detail, see **Creating a CDL Data Comparison Job**.

2.2 CDL User Permission Management

Scenario

Before using the CDL service, a cluster administrator needs to create a user and grant operation permissions to the user to meet service requirements.

CDL users are classified into administrators and common users. The default CDL user group for administrators and common users is **cdladmin** and **cdl**, respectively.

- Users associated with the **cdladmin** user group can perform any CDL operations.
- Users associated with the **cdl** user group can perform creation and query operations on CDL.

If Ranger authentication is enabled and you need to configure the creation, execution, query, or deletion permission for CDL users, see **Adding a Ranger Access Permission Policy for CDL**.

If Ranger authentication is manually disabled for a cluster, enable Ranger authentication by referring to **Enabling Ranger Authentication for MRS Cluster Services**.

NOTE

This section applies only to clusters with Kerberos authentication enabled.

Procedure

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **System**. On the navigation pane on the left, choose **Permission** > **User** and click **Create**.

Step 3 Enter a username, for example, **cdl_admin**.

Step 4 Set **User Type** to **Human-Machine**.

Step 5 Set **Password** and confirm your password.

Step 6 Set **User Group** and **Primary Group**.

- CDL administrator permissions: Add the **cdladmin** user group and set it to the primary group.
- Common CDL user permissions: Add the **cdl** user group and set it to the primary group.

Step 7 Click **OK**.

----End

2.3 Creating a Data Synchronization Job with CDL

CDL supports data synchronization or comparison tasks in multiple scenarios. This section describes how to import data from PgSQL to Kafka on the CDLService WebUI of a cluster with Kerberos authentication enabled. For more CDL job examples, see **Creating a CDL Job**.

Prerequisites

- The CDL and Kafka services have been installed in a cluster and are running properly.
- Write-ahead logging is enabled for the PostgreSQL database. For details, see **Policy for Modifying Write-Ahead Logs in PostgreSQL Databases**.
- You have created a human-machine user, for example, **cdluser**, added the user to user groups **cdladmin** (primary group), **hadoop**, and **kafka**, and associated the user with the **System_administrator** role on FusionInsight Manager.

Procedure

Step 1 Log in to FusionInsight Manager as user **cdluser** (change the password upon the first login) and choose **Cluster** > **Services** > **CDL**. On the **Dashboard** page, click the hyperlink next to **CDLService UI** to go to the native CDL page.

Step 2 Choose **Link Management** and click **Add Link**. On the displayed dialog box, set parameters for adding the **pgsql** and **kafka** links by referring to the following tables.

Table 2-12 PgSQL data link parameters

Parameter	Example Value
Link Type	pgsql
Name	pgsqllink
Host	10.10.10.10
Port	5432
DB Name	testDB
User	user
Password	*Password of the user user*
Description	-

Table 2-13 Kafka data link parameters

Parameter	Example Value
Link Type	kafka
Name	kafkalink
Description	-

Step 3 After the parameters are configured, click **Test** to check whether the data link is normal.

After the test is successful, click **OK**.

Step 4 On the **Job Management** page, click **Add Job**. In the displayed dialog box, configure the parameters and click **Next**.

Specifically:

Parameter	Example Value
Name	job_pgsqltokafka
Desc	xxx

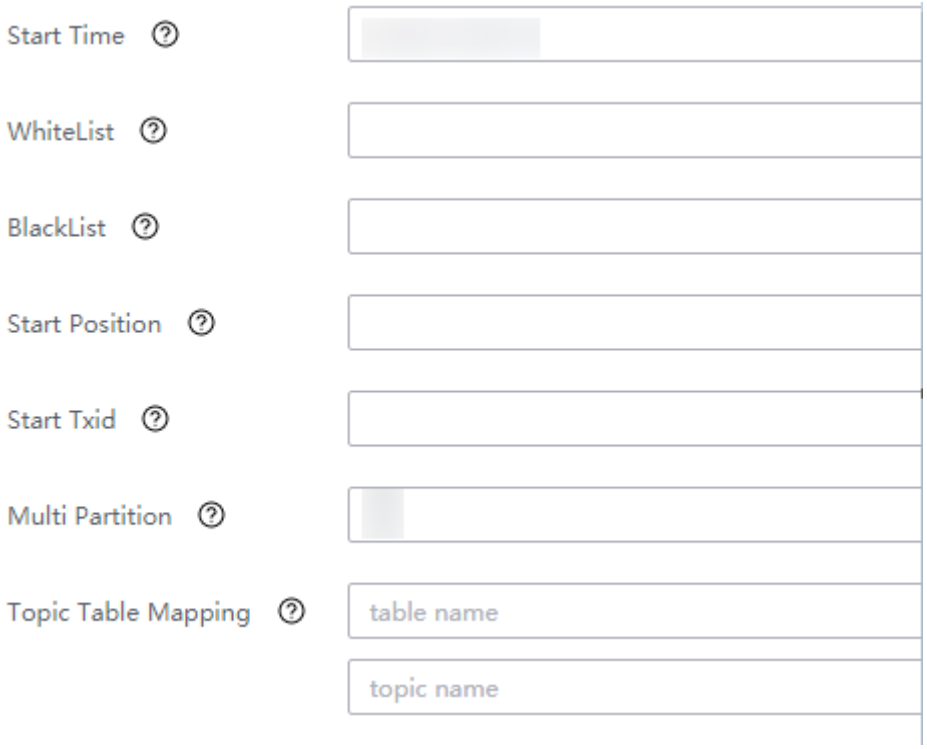
Step 5 Configure PgSQL job parameters.

1. On the **Job Management** page, drag the **pgsql** icon on the left to the editing area on the right and double-click the icon to go to the PgSQL job configuration page.

Table 2-14 PgSQL job parameters

Parameter	Example Value
Link	pgsqllink
Tasks Max	1
Mode	insert, update, and delete
Schema	public
dbName Alias	cdc
Slot Name	a4545sad
Slot Drop	No
Connect With Hudi	No
Use Exist Publication	Yes
Publication Name	test

2. Click the plus sign (+) to display more parameters.



NOTE

- **WhiteList**: Enter the name of the table in the database, for example, **myclass**.
- **Topic Table Mapping**: In the first text box, enter a topic name (the value must be different from that of **Name** in **Step 4**), for example, **myclass_topic**. In the second text box, enter a table name, for example, **myclass**. The value must be in one-to-one relationship with the topic name entered in the first text box.)

3. Click **OK**. The PgSQL job parameters are configured.

Step 6 Configure Kafka job parameters.

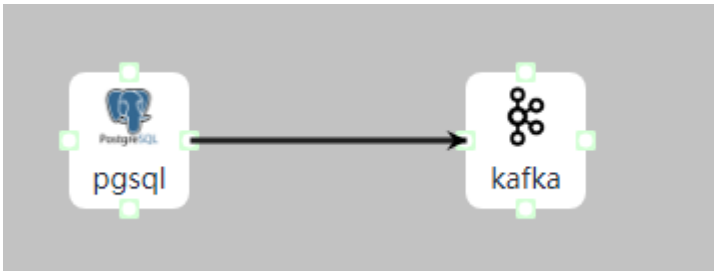
1. On the **Job Management** page, drag the **kafka** icon on the left to the editing area on the right and double-click the icon to go to the Kafka job configuration page. Configure parameters based on **Table 2-15**.

   Table 2-15 Kafka job parameter

Parameter	Example Value
Link	kafkalink

2. Click **OK**.

Step 7 After the job parameters are configured, drag the two icons to associate the job parameters and click **Save**. The job configuration is complete.



Step 8 In the job list on the **Job Management** page, locate the created jobs, click **Start** in the **Operation** column, and wait until the jobs are started.

Check whether the data transmission takes effect. For example, insert data into the table in the PgSQL database, go to the Kafka UI to check whether data is generated in the Kafka topic by referring to **Viewing Kafka Data Production and Consumption Details**.

----End

2.4 Preparing for Creating a CDL Job

2.4.1 Enabling Kafka High Reliability

Scenario

To execute the CDL data synchronization tasks listed in **Table 2-16**, enable the Kafka high reliability function to prevent data loss when Kafka is faulty or restarted.

Table 2-16 CDL tasks that use MRS Kafka to synchronize data

Data Source	Destination	Description
MySQL	Hudi	Synchronizes data from MySQL to Hudi.
	Kafka	Synchronizes data from MySQL to Kafka.
PgSQL	Hudi	Synchronizes data from PgSQL to Hudi.
	Kafka	Synchronizes data from PgSQL to Kafka.
ThirdKafka	Hudi	Synchronizes data from ThirdKafka to Hudi.
	Kafka	Synchronizes data from ThirdKafka to Kafka.
opengauss (supported by MRS 3.3.0 and later versions)	ThirdKafka (DMS/DRS) -> Hudi	Synchronizes data from GaussDB(for openGauss) to Hudi through ThirdKafka (DMS/DRS).
avro-oracle (supported by MRS 3.3.0 and later versions)	ThirdKafka (DMS/DRS) -> Hudi	Synchronizes data from avro-oracle to Hudi through ThirdKafka (DMS/DRS).

Prerequisites

- The CDL component has been installed in an MRS cluster and is running properly.
- CDL data synchronization tasks use the Kafka component.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. Click **Configurations** then **All Configurations**.

Step 2 Search for the parameters listed in **Table 2-17** in the search box in the upper right corner and change their values.

Table 2-17 Modifying Kafka parameters

Parameter	Recommended Value	Description
unclean.leader.election.enable	false	Whether a replica that is not in the ISR can be elected as the leader. If this parameter is set to **true**, data may be lost.
min.insync.replicas	2	Minimum number of replicas to which data is written when **offsets.commit.required.acks** is set to **-1**.

Step 3 Click **Save**.

Step 4 Choose **Dashboard**, click **More**, and select **Rolling-restart Service** to roll-restart Kafka.

----End

2.4.2 Logging In to the CDLService WebUI

Scenario

After CDL is installed in an MRS cluster, you can manage data connections and visualized jobs using the CDL web UI.

This section describes how to access the CDL web UI in an MRS cluster.

NOTE

You are advised to use Google Chrome to access the CDLService web UI because it is incompatible with Internet Explorer.

CDL cannot fetch tables whose names contain the dollar sign ($) and special Chinese characters.

Prerequisites

- The CDL component has been installed in an MRS cluster and is running properly.
- A user with the CDL management permission has been created for the cluster with Kerberos authentication enabled.

Procedure

Step 1 Log in to FusionInsight Manager as a user with the CDL management permissions or the **admin** user (for the cluster where Kerberos authentication is not enabled), and choose **Cluster** > **Services** > **CDL**.

Step 2 On the right of **CDLService UI**, click the link to access the CDLService web UI.

You can perform the following operations on the CDL web UI:

- **Driver Management**: You can upload, view, and delete a driver file corresponding to the connected database.
- **Link Management**: You can create, view, edit, and delete a data connection.
- **Job Management**: You can create, view, start, pause, restore, stop, restart, delete, or edit a job.
- **ENV Management**: You can create, view, edit, and delete Hudi environment variables.

----End

2.4.3 Uploading the Database Driver File

Scenario

CDL is a simple and efficient real-time data integration service. It captures events from various OLTP databases and pushes them to Kafka. When creating a database connection on the CDLService Web UI, you can upload the database's driver file to the Web UI for unified management.

Prerequisites

- You have obtained the driver JAR package of the database to be connected.
- You have downloaded the driver package of the MySQL or Oracle data source from their respective official website. **Table 2-18** lists the supported driver packages and their versions.

  Table 2-18 Drivers supported by MySQL and Oracle data sources

Data Source	Supported Driver Package
MySQL	mysql-connector-java-8.0.24.jar
Oracle	ojdbc8-12.2.0.1.jar

- A user with the CDL management permission has been created for the cluster with Kerberos authentication enabled. For details, see **CDL User Permission Management**.

Procedure

Step 1 Access the CDLService web UI as a user with the CDL management permissions or the **admin** user (for the cluster where Kerberos authentication is not enabled). For details, see **Logging In to the CDLService WebUI**.

Step 2 Choose **Driver Management** and click **Upload Driver**. In the displayed dialog box, select the prepared database driver file and click **Open**.

Step 3 On the **Driver Management** page, check whether the list of driver file names is displayed properly.

NOTE

- If a driver is no longer used or is mistakenly uploaded, click **Delete** to delete its driver file.
- If there are a large number of driver files, you can enter a driver file name in the search box to quickly search for the desired driver file.

----End

2.4.4 Creating a CDL Database Connection

Scenario

Create a database link on the CDLService web UI.

Prerequisites

- You have obtained the driver JAR package of the data to be connected.
- A user with the CDL management permission has been created for the cluster with Kerberos authentication enabled. For details, see **CDL User Permission Management**.

Procedure

Step 1 Access the CDLService web UI as a user with the CDL management permissions or the **admin** user (for the cluster where Kerberos authentication is not enabled). For details, see **Logging In to the CDLService WebUI**.

Step 2 Choose **Link Management** and click **Add Link**. In the displayed dialog box, enter the link name (cannot be the same as an existing one) and select the link type.

Step 3 Set other link parameters based on the link type.

Table 2-19 MySQL data link parameters

Parameter	Description	Example Value
Link Type	Link type	mysql
Name	Link name	mysqllink
DB driver	Uploaded MySQL driver file **mysql-connector-java-8.0.24.jar**	mysql-connector-java-8.0.24.jar
Host	IP address of the MySQL database	10.10.10.10
Port	MySQL database port	3306
User	User for accessing the MySQL database	user
Password	Password for accessing the MySQL database	*Password of the user user*

Parameter	Description	Example Value
Description	Data link description.	xxx

Table 2-20 PgSQL data link parameters

Parameter	Description	Example Value
Link Type	Link type	pgsql
Name	Link name	pgsqllink
Host	IP address of the PgSQL database	10.10.10.10
Port	PgSQL database port	5432
DB Name	PgSQL database name	testDB
User	User for accessing the PgSQL database	user
Password	Password for accessing the PgSQL database	*Password of the user user*
Description	Data link description	xxx

Table 2-21 Kafka data link parameters

Parameter	Description	Example Value
Link Type	Link type.	kafka
Name	Link name.	kafkalink
Description	Data link description.	-

Table 2-22 Hudi data link parameters

Parameter	Description	Example Value
Link Type	Link type.	hudi
Name	Link name.	hudilink
Storage Type	Storage type, which can be either of the following: **hdfs**: Data is stored in HDFS.	hdfs

Parameter	Description	Example Value
Auth KeytabFile	Keytab file of a user. You can click **Upload File** to upload the keytab file. Set this parameter only for a cluster in security mode. **NOTE** To obtain this file, log in to FusionInsight Manager and choose **System**. On the navigation pane on the left, choose **Permission** > **User** and choose **More** > **Download User Credential** in the **Operation** column.	${BIGDATA_HOME}/FusionInsight_CDL_*X.X.X*/install/FusionInsight-CDL-*X.X.X*/cdl/keytabs/cdl.keytab
Principal	Domain name of the user who accesses HDFS. Set this parameter only for a cluster in security mode.	cdl/test.com@HADOOP.COM
Description	Data link description.	xxx

Table 2-23 thirdparty-kafka data link parameters

Parameter	Description	Example Value
Link Type	Link type	thirdparty-kafka
Name	Link name	thirdparty-kafkalink
Bootstrap Servers	Kafka proxy instance, which can be set to a value in the format of *Service IP address of the Kafka Broker instance*:*Kafka port number*. **NOTE** If MRS Kafka is used as the source of thirdparty-kafka, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, click **Configuration**, search for the port in the search box, and obtain the port number based on the encryption protocol.	10.10.10.10:21005

Parameter	Description	Example Value
Security Protocol	Encryption protocol. Value options are as follows: • **SASL_PLAINTEXT** • **PLAINTEXT** • **SASL_SSL** • **SSL**	SASL_SSL
Username	Username specified when **SASL_SSL** is enabled during instance creation **NOTE** This parameter is available only when **Security Protocol** is set to **SASL_PLAINTEXT** or **SASL_SSL**.	test
Password	Password configured when **SASL_SSL** is enabled during instance creation **NOTE** This parameter is available only when **Security Protocol** is set to **SASL_PLAINTEXT** or **SASL_SSL**.	xxx
SSL Truststore Location	Path where the **client.truststore.jks** authentication file is stored **NOTE** This parameter is available only when **Security Protocol** is set to **SASL_SSL** or **SSL**.	-
SSL Truststore Password	Password of the **client.truststore.jks** certificate file **NOTE** This parameter is available only when **Security Protocol** is set to **SASL_SSL** or **SSL**.	xxx

Parameter	Description	Example Value
Datastore Type	Type of the upper-layer source. Value options are as follows: • MRS 3.2.0: – opengauss – ogg – oracle – drs-avro-oracle • MRS 3.3.0 and later versions: – drs-opengauss-json – ogg-oracle-avro – drs-oracle-json – drs-oracle-avro	• opengauss • drs-opengauss-json
DB driver	Uploaded thirdparty-kafka driver file **NOTE** This parameter is displayed when **Datastore Type** is set to **ogg** (or **ogg-oracle-avro** for MRS 3.3.0 and later versions).	-
Host	IP address of the thirdparty-kafka database **NOTE** This parameter is not displayed when **Datastore Type** is set to **oracle** (or **drs-oracle-json** for MRS 3.3.0 and later versions).	11.11.xxx.xxx,12.12.xxx.xxx
Port	thirdparty-kafka database port **NOTE** This parameter is not displayed when **Datastore Type** is set to **oracle** (or **drs-oracle-json** for MRS 3.3.0 and later versions).	8000
DB Name	thirdparty-kafka database name **NOTE** This parameter is displayed when **Datastore Type** is set to **opengauss** (or **drs-opengauss-json** for MRS 3.3.0 and later versions).	opengaussdb

Parameter	Description	Example Value
User	thirdparty-kafka database access user **NOTE** This parameter is not displayed when **Datastore Type** is set to **oracle** (or **drs-oracle-json** for MRS 3.3.0 and later versions).	opengaussuser
DB Password	Password for accessing the thirdparty-kafka database **NOTE** This parameter is not displayed when **Datastore Type** is set to **oracle** (or **drs-oracle-json** for MRS 3.3.0 and later versions).	*Password of the opengaussuser user*
Sid	Service ID of Oracle **NOTE** This parameter is displayed when **Datastore Type** is set to **ogg** (or **ogg-oracle-avro** for MRS 3.3.0 and later versions).	-
Description	Data link description.	-

Table 2-24 DWS data link parameters

Parameter	Description	Example Value
Link Type	Link type	dws
Name	Link name	dwslink
Host	IP address of the DWS database to be connected	10.10.10.10
Port	Database port	8000
DB Name	Name of the database to be connected to	default
User	Database access user	test
Password	Password for accessing the database	xxx
Description	Data link description.	-

Table 2-25 opengauss data link parameters

Parameter	Description	Example Value
Link Type	Link type	opengauss
Name	Link name	opengausslink
Host	IP address of the opengauss database to be connected	10.10.10.10
Port	Database port	8000
DB Name	Name of the database to be connected to	default
User	Database access user	test
Password	Password for accessing the database	xxx
Description	Data link description.	-

Table 2-26 ClickHouse data link parameters

Parameter	Description	Example Value
Link Type	Link type	clickhouse
Name	Link name	clickhouselink
Host	*Service IP address of the ClickHouseBalancer instance of ClickHouse:HTTP balancer port number*. Multiple ClickHouse instances can be connected using commas (,). **NOTE** To obtain the HTTP balancer port number, log in to Manager, choose **Cluster** > **Services** > **ClickHouse**, click **Logic Cluster**, and obtain the port number from the **HTTP Balancer Port** column in the cluster list.	10.10.10.10:21428
User	Database access user	test
Password	Password for accessing the database	xxx

Parameter	Description	Example Value
Description	Data link description.	-

Step 4 After the parameters are configured, click **Test** to check whether the data link is normal.

After the test is successful, click **OK**.

----End

2.4.5 Configuring CDL ENV Variables

Scenario

To capture data to or from Hudi, create and manage Hudi environment variables by performing the operations in this section.

Prerequisites

A user with the CDL management permission has been created for the cluster with Kerberos authentication enabled. For details, see **CDL User Permission Management**.

Procedure

Step 1 Access the CDLService web UI as a user with the CDL management permissions or the **admin** user (for the cluster where Kerberos authentication is not enabled). For details, see **Logging In to the CDLService WebUI**.

Step 2 Choose **ENV Management** and click **Add Env**. In the displayed dialog box, set related parameters.

Table 2-27 Parameters for adding an ENV

Parameter	Description	Example Value
Name	ENV name	spark-env
Type	ENV type	spark
Driver Memory	Memory for the driver process, in GB (by default).	1 GB
Executor Memory	Memory size for each Executor process, in GB by default. Its string format is the same as that of JVM.	1 GB

Parameter	Description	Example Value
Executor Cores	Number of CPU cores occupied by each Executor	1
Number Executors	Number of Executors	1
Queue	Name of the Yarn tenant queue. Jobs are submitted to the default queue if this parameter is not specified.	-
Description	ENV description	-

Step 3 Click **OK**.

After the ENV is created, you can click **Edit** or **Delete** in the **Operation** column to edit or delete the ENV, respectively.

----End

2.4.6 Configuring the Source Data Heartbeat Table for Data Integrity Check

Scenario

The heartbeat and data consistency check function is used to collect full-link information about CDL synchronization tasks, including the time required for sending data from the database management system RDBMS to Kafka, the time required for writing data from Kafka to Hudi, and the number of data records, and writes the data to a specific topic (cdl_snapshot_topic). You can consume the data in the topic and write the data to a specific Hudi table for data consistency check. The heartbeat data can be used not only to determine whether data before the heartbeat time has been synchronized to the data lake, but also to determine the data latency based on the transaction time, Kafka write time, data import start time, and data import end time.

In addition, for PgSQL tasks, configuring a heartbeat table can periodically push forward the LSN information recorded by the slot in the PgSQL database. This prevents database log stacking caused by the configuration of some tables with little changes in a task.

Configuring the Heartbeat Table for Capturing Data from Oracle GoldenGate (OGG) to a Hudi Job

Step 1 Run the following commands in the Oracle database where data needs to be synchronized to create a heartbeat table. The heartbeat table belongs to the **CDC_CDL** schema, the table name is **CDC_HEARTBEAT**, and the primary key is **CDL_JOB_ID**.

CREATE TABLE "CDC_CDL"."CDC_HEARTBEAT" (

"CDL_JOB_ID" VARCHAR(22) PRIMARY KEY,

"CDL_LAST_HEARTBEAT" TIMESTAMP,

SUPPLEMENTAL LOG DATA (ALL) COLUMNS

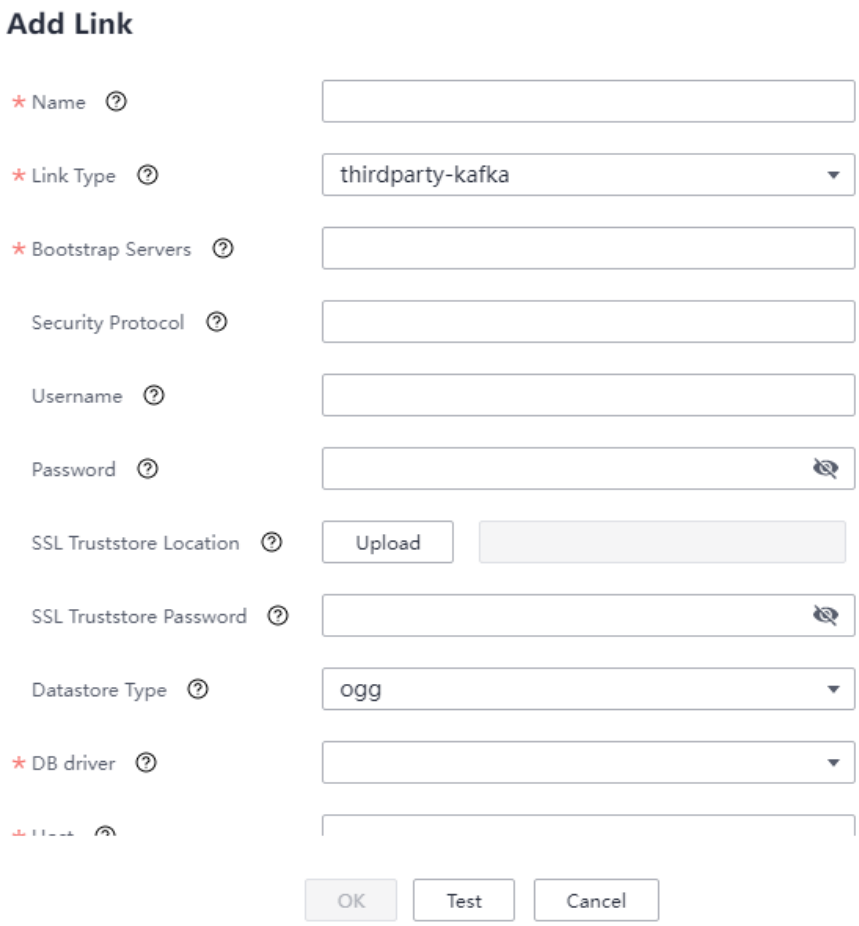
);

Step 2 Add the **CDC_HEARTBEAT** table to the Oracle or OGG job to ensure that heartbeat data can be properly sent to Kafka.

NOTE

For an Oracle job, go to **Step 4**.

Step 3 Configure the thirdparty-kafka (ogg) link on the CDL web UI and add the Oracle link information.



Step 4 After the configuration is complete, create a job for capturing data from OGG to Hudi on the CDL web UI and start the job to receive heartbeat data.

----End

Configuring the Heartbeat Table for Capturing Data from PostgreSQL to a Hudi Job

Step 1 Run the following commands in the PostgreSQL database to be synchronized to create a heartbeat table. The heartbeat table belongs to the **cdc_cdl** schema, the table name is **cdc_heartbeat**, and the primary key is **cdl_job_id**.

```
DROP TABLE IF EXISTS cdc_cdl.cdc_heartbeat;

CREATE TABLE cdc_cdl.cdc_heartbeat (

cdl_job_id int8 NOT NULL,

cdl_last_heartbeat timestamp(6)

);

ALTER TABLE cdc_cdl.cdc_heartbeat ADD CONSTRAINT cdc_heartbeat_pkey
PRIMARY KEY (cdl_job_id);
```

Step 2 After the heartbeat table is created, create a job for capturing data from PostgreSQL to Hudi on the CDL web UI and start the job to receive heartbeat data.

----End

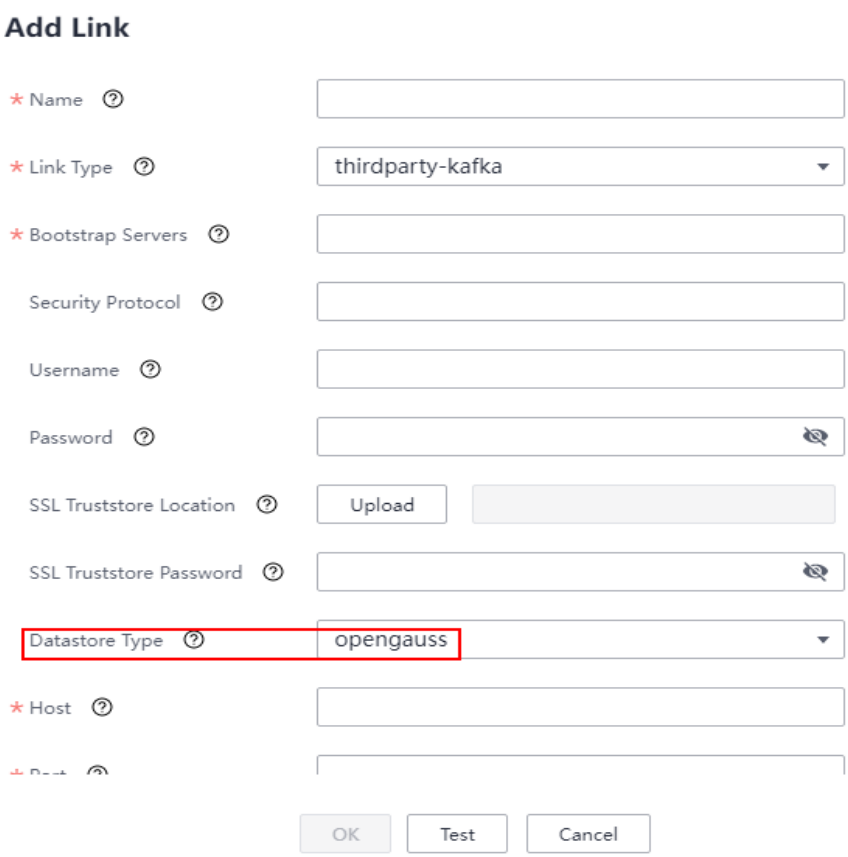
Configuring the Heartbeat Table from openGauss to a Hudi Job

Step 1 Run the following commands in the openGauss database to be synchronized to create a heartbeat table. The heartbeat table belongs to the **cdc_cdl** schema, the table name is **cdc_heartbeat**, and the primary key is **cdl_job_id**.

```
DROP TABLE IF EXISTS cdc_cdl.cdc_heartbeat;

CREATE TABLE cdc_cdl.cdc_heartbeat (

cdl_job_id int8 NOT NULL,

cdl_last_heartbeat timestamp(6)

);

ALTER TABLE cdc_cdl.cdc_heartbeat ADD CONSTRAINT cdc_heartbeat_pkey
PRIMARY KEY (cdl_job_id);
```

Step 2 Add the heartbeat table to the DRS job to ensure that the heartbeat table data is properly sent to the DRS Kafka.

Step 3 On the CDL web UI, add the openGauss link information when configuring the thirdparty-kafka link of openGauss. If one primary openGauss node and multiple standby openGauss nodes are deployed, enter all IP addresses in **Host**.



Step 4 After the configuration is complete, create a job for capturing data from thirdparty-kafka to Hudi on the CDL web UI and start the job to receive heartbeat data.

----End

Fields in a Data Consistency Check Message

Table 2-28 Fields in a data consistency check message

Field	Description
cdl_job_name	The name of the synchronization task to which the data in this batch belongs.
target_table_schema	The name of the schema to which the data in this batch is written.
target_table_name	The name of the Hudi table to which the data in this batch is written.
target_table_path	The path of the Hudi table to which the data in this batch is saved.
total_num	The total number of data records in this batch.

Field	Description
cdl_original_heartbeat	The maximum duration of heartbeat data in this batch. If this batch does not contain heartbeat data, the value is empty.
cdl_last_heartbeat	The minimum duration of heartbeat data in this batch. If this batch does not contain heartbeat data, the value of **event_time_min** is used.
insert_num	The total number of data **insert** events in this batch.
update_num	The total number of data **update** events in this batch.
delete_num	The total number of data **delete** events in this batch.
event_time_min	The minimum transaction submission time of the data source in this batch.
event_time_max	The maximum transaction submission time of the data source in this batch.
event_time_avg	The average transaction submission time of the data source in this batch.
kafka_timestamp_min	The minimum time for sending data in this batch to Kafka.
kafka_timestamp_max	The maximum time for sending data in this batch to Kafka.
begin_time	The time when the data in this batch starts to be written to Hudi.
end_time	The time when the data in this batch stops to be written to Hudi.
cdc_partitioned_time	The time partition field in the heartbeat table.
cdc_last_update_date	The time when the check record is written.

2.5 Creating a CDL Job

2.5.1 Creating a CDL Data Synchronization Job

Scenario

The CDLService web UI provides a visualized page for users to quickly create CDL jobs and import real-time data into the data lake.

Prerequisites

A user with the CDL management permission has been created for the cluster with Kerberos authentication enabled.

Procedure

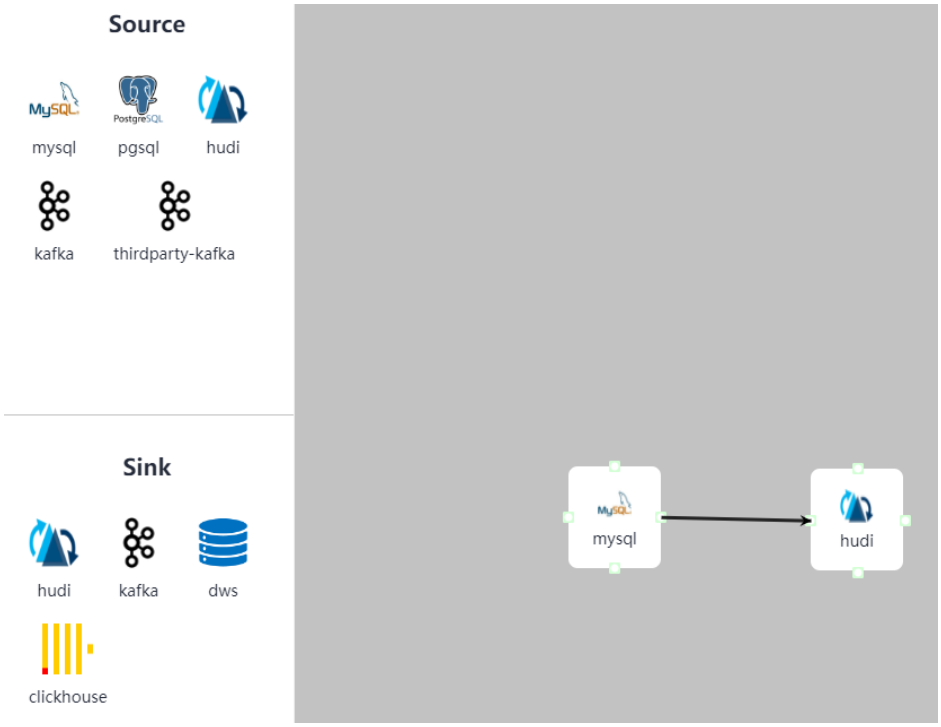
Step 1 Access the CDLService web UI as a user with the CDL management permissions or the **admin** user (for the cluster where Kerberos authentication is not enabled). For details, see **Logging In to the CDLService WebUI**.

Step 2 Choose **Job Management** > **Data synchronization task** and click **Add Job**. In the displayed dialog box, set related job parameters and click **Next**.

Parameter	Description	Example Value
Name	Job name	job_pgsqltokafka
Desc	Job description	xxx

Step 3 On the **Job Management** page, select and drag the target element from **Source** and **Sink** to the GUI on the right.

MRS 3.2.0:



MRS 3.3.0 and later versions:

Double-click the two elements to connect them and set related parameters as required.

To delete an element, select the element to be deleted and click **Delete** in the lower right corner of the page.

Table 2-29 MySQL job parameters

Parameter	Description	Example Value
Link	Created MySQL link	mysqllink
Tasks Max	Maximum number of tasks that can be created by a connector. For a connector of the database type, this parameter must be set to **1**.	1

Parameter	Description	Example Value
Mode	Type of the CDC event to be captured by the job. Value options are as follows: • **insert** • **update** • **delete**	insert, update, and delete
DB Name	MySQL database name	cdl-test
Schema Auto Create	Whether to create table schemas after the job is started	No
Connect With Hudi	Whether to connect to Hudi	Yes
DBZ Snapshot Locking Mode	Lock mode used when a task starts to execute a snapshot. Value options are as follows: • **minimal**: A global read lock is held only when the database schema and other metadata are obtained. • **extend**: A global read lock is held during the entire snapshot execution process, blocking all write operations. • **none**: No lock mode. The schema cannot be changed when a CDL task is started. Optional. Click  to display this parameter.	none
WhiteList	Whitelisted tables to be captured. Separate multiple tables using commas (,). Wildcards are supported. (Optional) This parameter is displayed when you click  .	testtable

Parameter	Description	Example Value
BlackList	Whitelisted tables not to be captured. Separate multiple tables using commas (,). Wildcards are supported. (Optional) This parameter is displayed when you click  .	-

Parameter	Description	Example Value
Multi Partition	Whether to enable multi-partition mode for topics. If enabled, you need to set **Topic Table Mapping** and specify the number of topic partitions, and the data of a single table will be scattered in multiple partitions. (Optional) This parameter is displayed when you click ⊞. **NOTE** • The data receiving sequence cannot be ensured. Exercise caution when setting this parameter. • The default number of partitions is 5. To change the number of partitions, log in to FusionInsight Manager, choose **Cluster** > **Services** > **CDL**, click **Configurations**, search for **topics.max.partitions** in the search box, and change the value to the number of partitions to be changed. For example: Change the value to 10, save the configuration, and restart the CDL service. • In MRS 3.3.0 and later versions, if the source table is a partitioned table and this parameter is set to **No**, the number of topic partitioned tables created using CDL is the number of source table partitions plus 1.	No

Parameter	Description	Example Value
Topic Table Mapping	Mapping between topics and tables. If configured, table data can be sent to the specified topic. If multi-partitioning is enabled, you need to set the number of partitions, which must be greater than 1. In MRS 3.3.0 and later versions, when the time of the source data is earlier than the specified data filtering time, the data is discarded. When the time of the source data is later than the specified data filtering time, the data is sent to the downstream. This parameter is displayed when you click  . This parameter is mandatory if **Connect With Hudi** is set to **Yes**.	testtable testtable_topic 2023/03/10 11:33:37 (supported in MRS 3.3.0 and later versions)

Table 2-30 PgSQL job parameters

Parameter	Description	Example Value
Link	Created PgSQL link.	pgsqllink
Tasks Max	Maximum number of tasks that can be created by a connector. For a connector of the database type, this parameter must be set to **1**.	1
Mode	Type of the CDC event to be captured by the job. The options are as follows: • **insert** • **update** • **delete**	insert, update, and delete

Parameter	Description	Example Value
dbName Alias	Database name.	test
Schema	Schema of the database to be connected to.	public
Slot Name	Name of the PostgreSQL logical replication slot. The value can contain lowercase letters and underscores (_), and cannot be the same in any other job.	test_solt
Enable FailOver Slot	Whether to enable the failover slot function. After it is enabled, the information about the logical replication slot specified as the failover slot is synchronized from the active instance to the standby instance. In this manner, logical subscription can continue even upon an active/standby switchover, implementing the failover of the logical replication slot.	No
Slot Drop	Whether to delete the slot when a task is stopped	No
Connect With Hudi	Whether to connect to Hudi.	Yes
Use Exist Publication	Use a created publication	Yes
Publication Name	Name of a created publication This parameter is available when **Use Exist Publication** is set to **Yes**.	test
Start Time (MRS 3.2.0)	Start time for synchronizing tables	2022/03/16 11:33:37
Data Filter Time (MRS 3.3.0 and later versions)	Start time of data filtering	2022/03/16 11:33:37

Parameter	Description	Example Value
WhiteList	Whitelisted tables to be captured. Separate multiple tables using commas (,). Wildcards are supported. (Optional) This parameter is displayed when you click ⊞.	testtable
BlackList	Whitelisted tables not to be captured. Separate multiple tables using commas (,). Wildcards are supported. (Optional) This parameter is displayed when you click ⊞.	-
Start Position	Start LSN of the data captured by a task	-
Start Txid	Start TXID of the data captured by a task	-

Parameter	Description	Example Value
Multi Partition	Whether to enable multi-partition mode for topics. If enabled, you need to set **Topic Table Mapping** and specify the number of topic partitions, and the data of a single table will be scattered in multiple partitions. (Optional) This parameter is displayed when you click ⊞. **NOTE** • The data receiving sequence cannot be ensured. Exercise caution when setting this parameter. • The default number of partitions is 5. To change the number of partitions, log in to FusionInsight Manager, choose **Cluster** > **Services** > **CDL**, click **Configurations**, search for **topics.max.partitions** in the search box, and change the value to the number of partitions to be changed. For example: Change the value to 10, save the configuration, and restart the CDL service. • In MRS 3.3.0 and later versions, if the source table is a partitioned table and this parameter is set to **No**, the number of topic partitioned tables created using CDL is the number of source table partitions plus 1.	No

Parameter	Description	Example Value
Topic Table Mapping	Mapping between topics and tables. If configured, table data can be sent to the specified topic. If multi-partitioning is enabled, you need to set the number of partitions, which must be greater than 1. In MRS 3.3.0 and later versions, when the time of the source data is earlier than the specified data filtering time, the data is discarded. When the time of the source data is later than the specified data filtering time, the data is sent to the downstream. This parameter is displayed when you click ⊞. This parameter is mandatory if **Connect With Hudi** is set to **Yes**.	testtable testtable_topic

Table 2-31 Source Hudi job parameters (MRS 3.2.0)

Parameter	Description	Example Value
Link	Link used by the Hudi app	hudilink
Interval	Interval for synchronizing the Hudi table, in seconds	10
Start Time	Start time for synchronizing tables	2022/03/16 11:40:52
Max Commit Number	Maximum number of commits that can be pulled from an incremental view at a time.	10

Parameter	Description	Example Value
Hudi Custom Config	Customized configuration related to Hudi.	-
Table Info	Detailed configuration information about the synchronization table. Hudi and DWS must have the same table names and field types.	{"table1": [{"source.database":"base1","source.tablename":"table1"}],"table2": [{"source.database":"base2","source.tablename":"table2"}],"table3": [{"source.database":"base3","source.tablename":"table3"}]}
Execution Env	Environment variable required for running the Hudi App. If no ENV is available, manually create one.	defaultEnv

Table 2-32 Source Hudi job parameters (MRS 3.3.0 and later versions)

Parameter	Description	Example Value
Link	Link used by the Hudi app	hudilink
Interval	Interval for synchronizing the Hudi table, in seconds	10
Data Filter Time	Data filtering time	2023/08/16 11:40:52
Max Commit Number	Maximum number of commits that can be pulled from an incremental view at a time.	10
Configuring Hudi Table Attributes	View for configuring attributes of the Hudi table. The value can be: ● Visual View ● JSON View	Visual View
Hudi Custom Config	Customized configuration related to Hudi.	-

Parameter	Description	Example Value
Table Info	Detailed configuration information about the synchronization table. Hudi and DWS must have the same table names and field types.	{"table1":[{"source.database":"base1","source.tablename":"table1"}],"table2":[{"source.database":"base2","source.tablename":"table2"}],"table3":[{"source.database":"base3","source.tablename":"table3"}]}
Table Info-Section Name	Label name of a single table. Only digits, letters, and underscores (_) are supported.	-
Table Info-Source DataBase	Name of the Hudi database to be synchronized	base1
Table Info-Source TableName	Name of the Hudi table to be synchronized	table1
Table Info-Target SchemaName	Schema name of the target database to which data is written	-
Table Info-Target TableName	Table name of the target database to which data is written	-
Table Info-Enable Sink Precombine	Whether to enable pre-combine for the target database. Currently, the pre-combine function can be enabled only when the target database is DWS. This function is used to overwrite existing data at the target when new pre-combine fields are more than the target pre-combine fields. If the new pre-combine fields are less than the target pre-combine fields, new data is discarded	Yes
Table Info-Custom Config	Hudi custom configuration	-

Parameter	Description	Example Value
Execution Env	Environment variable required for running the Hudi App. If no ENV is available, manually create one.	defaultEnv

Table 2-33 Source Kafka job parameters (Applicable only to MRS 3.2.0)

Parameter	Description	Example Value
Link	Created Kafka link	kafkalink

Table 2-34 thirdparty-kafka job parameters

Parameter	Description	Example Value
Link	Created thirdparty-kafka link	thirdparty-kafkalink
DB Name	Name of the database to be connected to. The name can contain only letters, digits, underscores (_), and hyphens (-), and must start with a letter.	opengaussdb
Schema	Schema of the database to be checked	oprngaussschema
Datastore Type	Type of the upper-layer source. Value options are as follows: • MRS 3.2.0: – opengauss – ogg – oracle – drs-avro-oracle • MRS 3.3.0 and later versions – drs-opengauss-json – ogg-oracle-avro – drs-oracle-json – drs-oracle-avro	• opengauss • drs-opengauss-json

Parameter	Description	Example Value
Avro Schema Topic	Schema topic used by OGG Kafka to store table schemas in JSON format. **NOTE** This parameter is available when **Datastore Type** is set to **ogg**.	ogg_topic
Source Topics	Source topics can contain letters, digits, and special characters (-,_). Topics must be separated by commas (,).	topic1
Tasks Max	Maximum number of tasks that can be created by a connector. For a connector of the database type, this parameter must be set to **1**.	10
Tolerance	Fault tolerance policy. ● **none**: indicates low tolerance and the Connector task will fail if an error occurs. ● **all**: indicates high tolerance and all failed records will be ignored if an error occurs.	all
Start Time (MRS 3.2.0)	Start time for synchronizing tables	2022/03/16 14:14:50
Data Filter Time (MRS 3.3.0 and later versions)	Start time of data filtering	2022/03/16 11:33:37
Kaka Message Format	Supported formats. The options are as follows: ● Debezium Json ● CDL Json **NOTE** ● Only MRS 3.3.0 and later versions support this parameter. ● This parameter is available only when **Datastore Type** is set to **drs-opengauss-json**.	CDL Json

Parameter	Description	Example Value
Multi Partition	Whether to enable multi-partitioning for topics. If it is enabled, you need to set **Topic Table Mapping** and specify the number of topic partitions, and the data of a single table will be scattered in multiple partitions. ● In MRS 3.2.0, if **Datastore Type** is set to **ogg** or **drs-avro-oracle** or **oracle**, the topic multi-partition function cannot be enabled. ● In MRS 3.3.0 and later versions, when **Datastore Type** is set to **ogg-oracle-avro** or **drs-oracle-avro** or **drs-oracle-json**, the topic multi-partition function cannot be enabled. **NOTE** The default number of partitions is 5. To change the number of partitions, log in to FusionInsight Manager, choose **Cluster** > **Services** > **CDL**, click **Configurations**, search for **topics.max.partitions** in the search box, and change the value to the number of partitions to be changed. For example: Change the value to 10, save the configuration, and restart the CDL service.	No

Parameter	Description	Example Value
Topic Table Mapping	Mapping between topics and tables. If configured, table data can be sent to the specified topic. If multi-partitioning is enabled, you need to set the number of partitions, which must be greater than 1. In MRS 3.3.0 and later versions, when the time of the source data is earlier than the specified data filtering time, the data is discarded. When the time of the source data is later than the specified data filtering time, the data is sent to the downstream.	testtable testtable_topic 2023/03/10 11:33:37 (MRS 3.3.0 and later versions)

Table 2-35 opengauss job parameters (applicable only to MRS 3.3.0 and later versions)

Parameter	Description	Example Value
Link	Created openGauss link	opengausslink
Tasks Max	Maximum number of tasks that can be created by the connector. The value is **1**.	1
Mode	Type of the CDC event to be captured by the job. Value options are as follows: • **insert** • **update** • **delete**	insert, update, and delete
dbName Alias	Database name.	test

Parameter	Description	Example Value
Slot Name	Name of the openGauss logical replication slot The value can contain lowercase letters and underscores (_), and cannot be the same in any other job.	test_solt
Slot Drop	Whether to delete the slot when a task is stopped	No
Connect With Hudi	Whether to connect to Hudi	Yes
WhiteList	Whitelisted tables that will be captured. Separate multiple tables using commas (,). Wildcards are supported.	testtable
Data Filter Time	Data filtering time	-

Parameter	Description	Example Value
Multi Partition	Whether to enable multi-partition mode for topics. If enabled, you need to set **Topic Table Mapping** and specify the number of topic partitions, and the data of a single table will be scattered in multiple partitions. **NOTE** • The data receiving sequence cannot be ensured if this function is enabled. Exercise caution when setting this parameter. • The default number of partitions is 5. To change the number of partitions, log in to FusionInsight Manager, choose **Cluster** > **Services** > **CDL**, click **Configurations**, search for **topics.max.partitions** in the search box, and change the value to the number of partitions to be changed. For example: Change the value to 10, save the configuration, and restart the CDL service.	No
Key Management Tool	Key management tool. Currently, only **his_kms** is supported.	his_kms
Key Environment Information	Key information This parameter is available only when **Key Management Tool** is configured.	-
Custom Config	Custom decoding configuration	-

Parameter	Description	Example Value
Topic Table Mapping	Mapping between topics and table. The format of the table name is *Schema name*.*Table name*. If configured, table data can be sent to the specified topic. If multi-partitioning is enabled, you need to set the number of partitions, which must be greater than 1. When the time of the source data is earlier than the specified data filtering time, the data is discarded. When the time of the source data is later than the specified data filtering time, the data is sent to the downstream. This parameter is mandatory if **Connect With Hudi** is set to **Yes**.	cdlschema.testtable testtable_topic 2023/03/10 11:33:37

Table 2-36 Sink Hudi job parameters

Parameter	Description	Example Value
Link	Created Hudi link.	hudilink
Path	Path for storing data.	/cdldata
Interval	Spark RDD execution interval, in seconds.	1
Max Rate Per Partition	Maximum rate for reading data from each Kafka partition using the Kafka direct stream API. It is the number of records per second. **0** indicates that the rate is not limited.	0
Parallelism	Parallelism for writing data to Hudi.	100

Parameter	Description	Example Value
Target Hive Database	Database of the target Hive	default
Configuring Hudi Table Attributes	View for configuring attributes of the Hudi table. The value can be: • Visual View • JSON View	Visual View
Global Configuration of Hudi Table Attributes	Global parameters on Hudi.	-
Configuring the Attributes of the Hudi Table	Configuration of the Hudi table attributes.	-
Configuring the Attributes of the Hudi Table: Table Name/ Source Table Name (MRS 3.3.0 and later versions)	Source table name	-
Configuring the Attributes of the Hudi Table: Table Type Opt Key	Hudi table type. The options are as follows: • COPY_ON_WRITE • MERGE_ON_READ	MERGE_ON_READ
Configuring the Attributes of the Hudi Table: Hudi TableName Mapping	Hudi table name. If this parameter is not set, the name of the Hudi table is the same as that of the source table by default.	-
Configuring the Attributes of the Hudi Table: Hive TableName Mapping	Mapping between Hudi tables and Hive tables.	-
Configuring the Attributes of the Hudi Table: Table Primarykey Mapping	Primary key mapping of the Hudi table	id

Parameter	Description	Example Value
Configuring the Attributes of the Hudi Table: Table Hudi Partition Type	Mapping between the Hudi table and partition fields. If the Hudi table uses partitioned tables, you need to configure the mapping between the table name and partition fields. The value can be **time** or **customized**.	time
Configuring the Attributes of the Hudi Table: Custom Config	Custom configuration **NOTE** In MRS 3.3.0 and later versions, when data is synchronized from MySQL to Hudi, you need to specify the **hoodie.datasource.write.precombine.field** parameter. If the timestamp and datetime of MySQL support only second-level precision, do not specify fields of timestamp or datetime type, or **_hoodie_event_time** as the pre-combine field. This function is used to overwrite existing data at Hudi when new pre-combine fields are more than Hudi pre-combine fields. If the new pre-combine fields are less than Hudi pre-combine fields, new data is discarded.	-
Execution Env	Environment variable required for running the Hudi App. If no ENV is available, create one by referring to **Configuring CDL ENV Variables**.	defaultEnv

Table 2-37 Sink Kafka job parameters

Parameter	Description	Example Value
Link	Created Kafka link	kafkalink

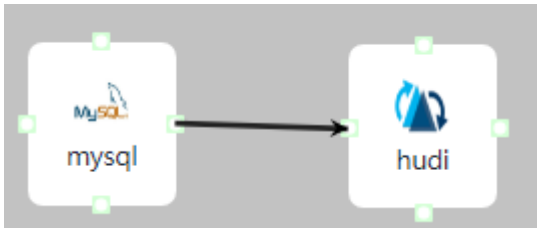
Table 2-38 DWS job parameters

Parameter	Description	Example Value
Link	Link used by Connector	dwslink
Query Timeout	Timeout interval for connecting to DWS, in milliseconds	180000
Batch Size	Amount of data batch written to DWS	50
Sink Task Number	Maximum number of concurrent jobs when a table is written to DWS.	-
DWS Custom Config	Custom configuration	-

Table 2-39 ClickHouse job parameters

Parameter	Description	Example Value
Link	Link used by Connector	clickhouselink
Query Timeout	Timeout interval for connecting to ClickHouse, in milliseconds	60000
Batch Size	Amount of data batch written to ClickHouse **NOTE** It is best practice to set this parameter to a large value. The recommended value range is 10000-100000.	100000

Step 4 After the job parameters are configured, drag the two icons to associate the job parameters and click **Save**. The job configuration is complete.



Step 5 In the job list on the **Job Management** page, locate the created jobs, click **Start** in the **Operation** column, and wait until the jobs are started.

Check whether the data transmission takes effect, for example, insert data into the table in the MySQL database and view the content of the file imported to Hudi.

----End

2.5.2 Creating a CDL Data Comparison Job

Scenario

Data comparison checks the consistency between data in the source database and that in the target Hive. If the data is inconsistent, CDL can attempt to repair the inconsistent data.

The current data comparison task supports manual full comparison. The data comparison task runs in On Yarn mode, and the comparison result is uploaded to HDFS directories.

NOTE

- Currently, only basic data types can be compared. Special data types such as date, timestamp, decimal, numeric, and JSON cannot be compared.
- Data cannot be compared for tables whose field names contain database keywords.
- A data comparison task for a single table supports comparison of a maximum of 100 fields. If a table contains more than 100 fields, you can specify two whitelists of different comparison fields for data comparison.
- Currently, only the data captured from PgSQL to Hudi can be compared. If the comparison result is inconsistent, a report address is generated only when there are no more than 2000 inconsistent data records. If there are more than 2000 inconsistent data records, no report address is generated and data cannot be repaired.
- If the Kafka lag of the CDL task involved in the comparison is not 0, the comparison result is inconsistent.

Prerequisites

1. You have prepared the Hive UDF JAR package, copied **${BIGDATA_HOME}/FusionInsight_CDL_*/install/FusionInsight-CDL-*/cdl/hive-checksum/cdl-dc-hive-checksum-*.jar** from the CDL installation directory to the **${BIGDATA_HOME}/third_lib/Hive** directory of Hive, and set the permission on the JAR package to **750** or higher.

   ```
   [root@192-168-42-72 /opt/huawei/Bigdata/third_lib/Hive ]#ll
   total 16
   -rwxrwxrwx. 1 root root 4783 Mar  3 09:52 cdl-dc-hive-checksum-1.0-SNAPSHOT.jar
   -rwxrwxrwx. 1 root root 4628 Mar  2 17:13 cdl_md5_xor_v4-1.0.jar
   [root@192-168-42-72 /opt/huawei/Bigdata/third_lib/Hive ]#
   ```

2. A user with the CDL management permission has been created for the cluster with Kerberos authentication enabled.. If Ranger authentication is enabled for the current cluster, grant the Hive administrator permission and UDF operation permission to the user by referring to **Adding a Ranger Access Permission Policy for Hive**.
3. You have created a global UDF algorithm on the Hive client as a user with the Hive administrator permission.

   Run the following command to create the **CheckSum** function in the default database:

create function checksum_aggregate as 'com.huawei.hive.checksum.ChecksumUdaf'

4. A CDL synchronization task exists. The comparison task determines the data to be compared based on the synchronization task status and data synchronization status.
5. The database user in the data synchronization task associated with data comparison task must have the **create function** permission on the current schema.

Procedure

Step 1 Log in to the CDLService web UI as the created user user (for the cluster where Kerberos authentication is not enabled). For details, see **Logging In to the CDLService WebUI**.

Step 2 Choose **Job Management** > **Data comparison task** and click **Add Job**. In the displayed dialog box, set related job parameters and click **Next**.

Parameter	Description	Example
Name	Name of the data comparison task.	job_dc_test
CDL Job Name	Name of the associated synchronization task. (Note: The user who runs the comparison task is the user of the Hudi Link in the associated synchronization task.)	pg2hudi_test
Execution Env	Environment variable required for running Spark tasks. If no ENV is available, create one by referring to **Configuring CDL ENV Variables**.	dc_env
Desc	Description of the task.	-

Step 3 On the **Create Compare-Pair** page, set related parameters and click **Create**.

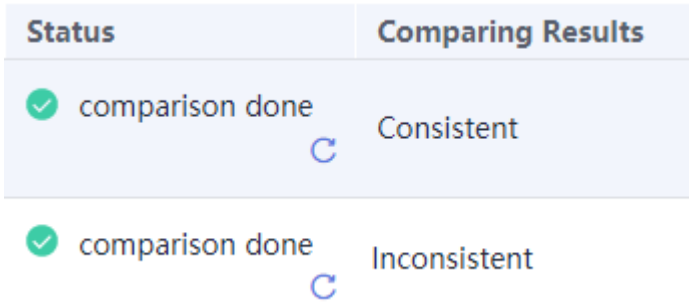
Parameter	Description	Example

Name	Name of the current comparison task.	test
Source Table	Source table name.	tabletest
Target Table	Target table name.	tabletest
WhiteList Columns	Column family involved in data comparison.	-
BlackList Columns	Column family not involved in data comparison.	-
Where Condition	User-defined comparison conditions.	-

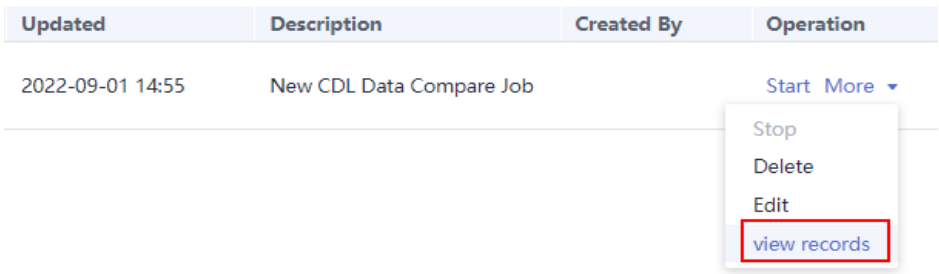
To compare multiple tables, click **Add**.

Step 4 In the data comparison task list, click **Start** in the row of the task to start data comparison.

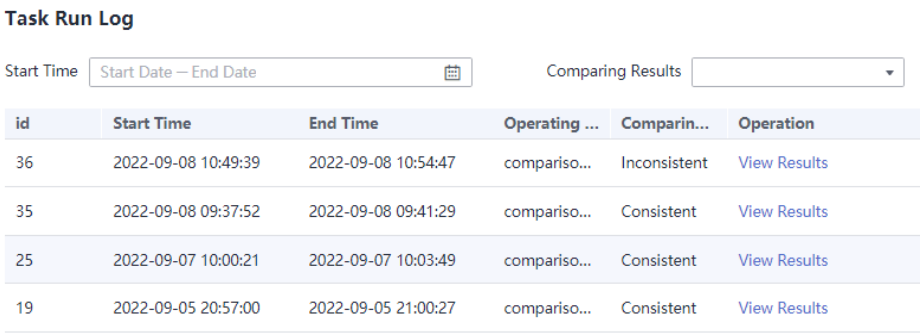
Step 5 After the execution is complete, view the result in the **Comparing Result** column.



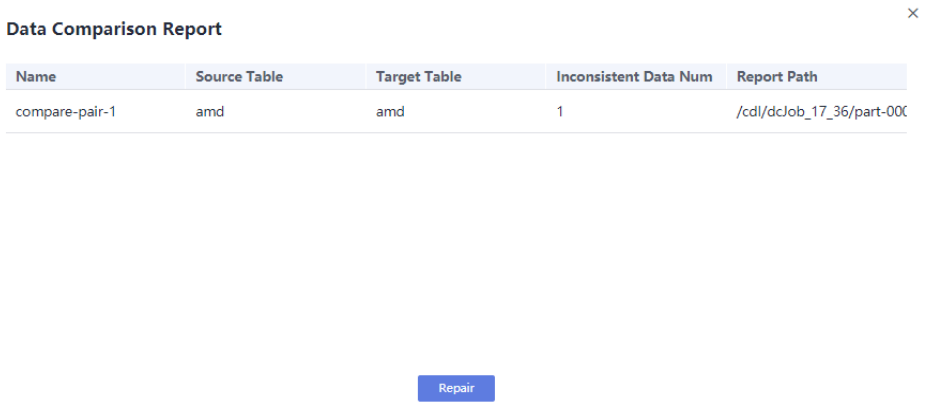
Step 6 If the result is **Inconsistent**, click **More** and select **view records**.



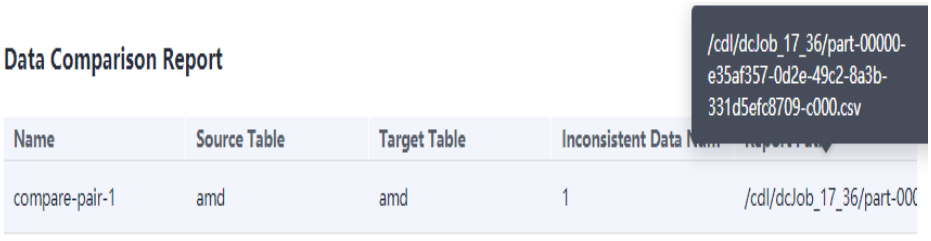
Step 7 In the **Task Run Log** window, locate the target task and click **View Results** in the **Operation** column.



Step 8 Click **Repair** to repair the data.



Step 9 After the repair is complete, check whether the value of **Comparing Result** is **Consistent**. If yes, data repair is successful. If not, the repair fails. In this case, obtain the report from the corresponding HDFS directory based on the value of **Report Path** and manually repair the data.



----End

2.5.3 Synchronizing Data from PgSQL to Kafka Using CDL

Scenario

This section describes how to import data from PgSQL to Kafka by using the CDLService web UI of a cluster with Kerberos authentication enabled in MRS 3.2.0.

Prerequisites

- The CDL and Kafka services have been installed in a cluster and are running properly.
- Write-ahead logging is enabled for the PostgreSQL database. For details, see **Policy for Modifying Write-Ahead Logs in PostgreSQL Databases**.
- You have created a human-machine user, for example, **cdluser**, added the user to user groups **cdladmin** (primary group), **hadoop**, and **kafka**, and associated the user with the **System_administrator** role on FusionInsight Manager.

Procedure

Step 1 Log in to FusionInsight Manager as user **cdluser** (change the password upon the first login) and choose **Cluster** > **Services** > **CDL**. On the **Dashboard** page, click the hyperlink next to **CDLService UI** to go to the native CDL page.

Step 2 Choose **Link Management** and click **Add Link**. On the displayed dialog box, set parameters for adding the **pgsql** and **kafka** links by referring to the following tables. **Creating a CDL Database Connection** describes the data link parameters.

Table 2-40 PgSQL data link parameters

Parameter	Example Value
Link Type	pgsql
Name	pgsqllink
Host	10.10.10.10
Port	5432
DB Name	testDB
User	user
Password	*Password of the user user*
Description	-

Table 2-41 Kafka data link parameters

Parameter	Example Value
Link Type	kafka
Name	kafkalink
Description	-

Step 3 After the parameters are configured, click **Test** to check whether the data link is normal.

After the test is successful, click **OK**.

Step 4 On the **Job Management** page, click **Add Job**. In the displayed dialog box, configure the parameters and click **Next**.

Specifically:

Parameter	Example Value
Name	job_pgsqltokafka
Desc	xxx

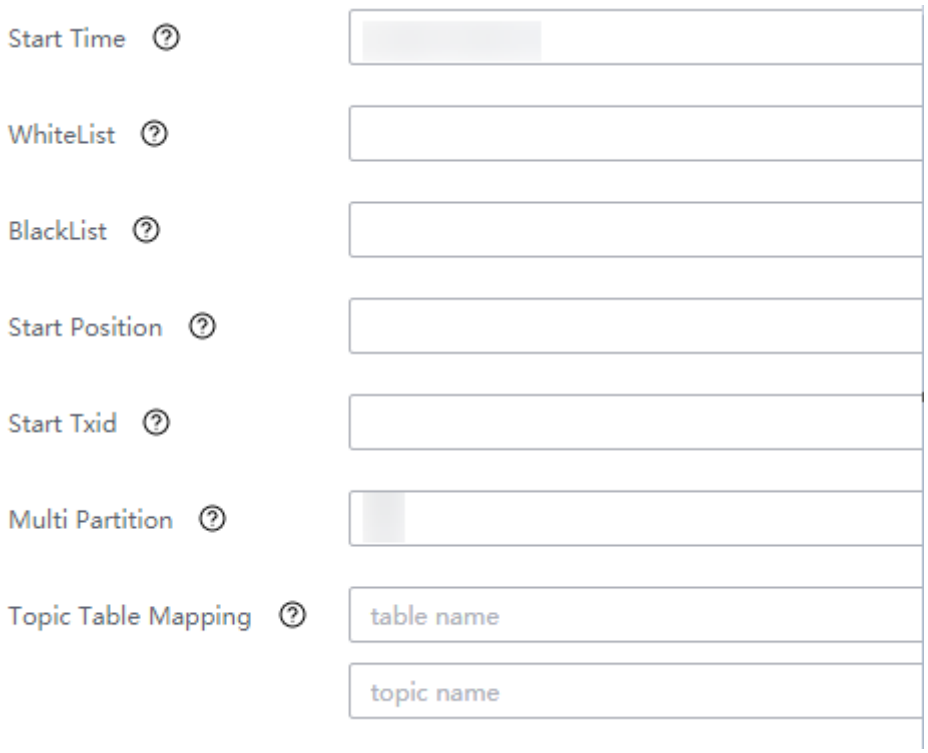
Step 5 Configure PgSQL job parameters.

1. On the **Job Management** page, drag the **pgsql** icon on the left to the editing area on the right and double-click the icon to go to the PgSQL job configuration page. For details about job parameters, see **Creating a CDL Data Synchronization Job**.

   Table 2-42 PgSQL job parameters

Parameter	Example Value
Link	pgsqllink
Tasks Max	1
Mode	insert, update, and delete
Schema	public
dbName Alias	cdc
Slot Name	test_slot
Slot Drop	No
Connect With Hudi	No
Use Exist Publication	Yes
Publication Name	test

2. Click the plus sign (+) to display more parameters.



NOTE

- **WhiteList**: Enter the name of the table in the database, for example, **myclass**.
- **Topic Table Mapping**: In the first text box, enter a topic name (the value must be different from that of **Name** in **Step 4**), for example, **myclass_topic**. In the second text box, enter a table name, for example, **myclass**. The value must be in one-to-one relationship with the topic name entered in the first text box.)

3. Click **OK**. The PgSQL job parameters are configured.

Step 6 Configure Kafka job parameters.

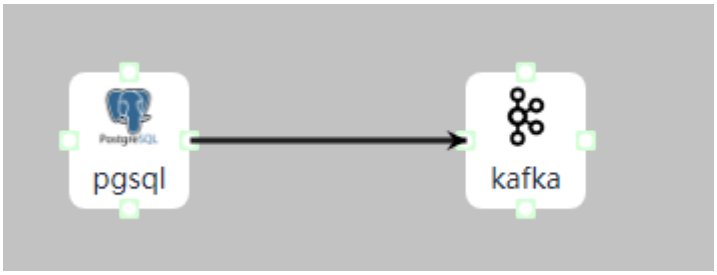
1. On the **Job Management** page, drag the **kafka** icon on the left to the editing area on the right and double-click the icon to go to the Kafka job configuration page. Configure parameters based on **Table 2-43**.

Table 2-43 Kafka job parameter

Parameter	Example Value
Link	kafkalink

2. Click **OK**.

Step 7 After the job parameters are configured, drag the two icons to associate the job parameters and click **Save**. The job configuration is complete.



Step 8 In the job list on the **Job Management** page, locate the created jobs, click **Start** in the **Operation** column, and wait until the jobs are started.

Check whether the data transmission takes effect. For example, insert data into the table in the PgSQL database, go to the Kafka UI to check whether data is generated in the Kafka topic by referring to **Viewing Kafka Data Production and Consumption Details**.

----End

2.5.4 Synchronizing Data from PgSQL to Hudi Using CDL

Scenario

This section describes how to import data from PgSQL to Hudi by using the CDLService web UI of a cluster with Kerberos authentication enabled in MRS 3.2.0.

Prerequisites

- The CDL and Hudi services have been installed in a cluster and are running properly.
- The prerequisites for the PgSQL database have been met. For details, see **Policy for Modifying Write-Ahead Logs in PostgreSQL Databases**.
- You have created a human-machine user, for example, **cdluser**, added the user to user groups **cdladmin** (primary group), **hadoop**, **kafka**, and **supergroup**, and associated the user with the **System_administrator** role on FusionInsight Manager.

Procedure

Step 1 Log in to FusionInsight Manager as user **cdluser** (change the password upon your first login), choose **Cluster** > **Services** > **CDL**, and click the link next to **CDLService UI** to go to the CDLService web UI.

Step 2 Choose **Link Management** and click **Add Link**. In the displayed dialog box, set parameters for adding the **pgsql** and **hudi** links by referring to the following tables. **Creating a CDL Database Connection** describes the data link parameters.

Table 2-44 PgSQL data link parameters

Parameter	Example
Link Type	pgsql

Parameter	Example
Name	pgsqllink
Host	10.10.10.10
Port	5432
DB Name	testDB
User	user
Password	*Password of the **user** user*
Description	-

Table 2-45 Hudi data link parameters

Parameter	Example
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	xxx

Step 3 After the parameters are configured, click **Test** to check whether the data link is normal.

After the test is successful, click **OK**.

Step 4 (Optional) Choose **ENV Management** and click **Add Env**. In the displayed dialog box, configure the parameters based on the following table.

Table 2-46 Parameters for adding an ENV

Parameter	Example Value
Name	test-env
Driver Memory	1 GB
Type	spark
Executor Memory	1 GB
Executor Cores	1
Number Executors	1

Parameter	Example Value
Queue	-
Description	-

Click **OK**.

Step 5 Choose **Job Management** > **Data synchronization task** and click **Add Job**. In the displayed dialog box, set parameters and click **Next**.

Parameters are as follows.

Parameter	Example
Name	job_pgtohudi
Desc	-

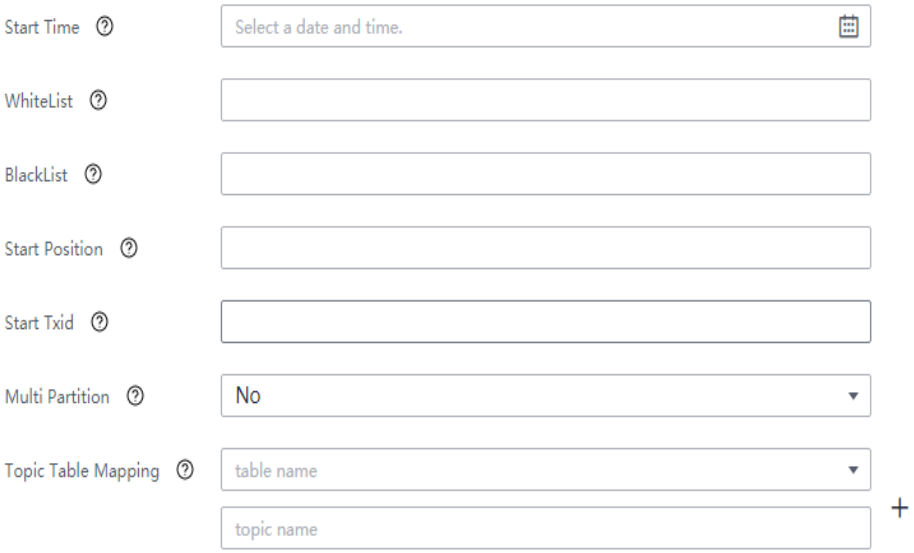
Step 6 Configure PgSQL job parameters.

1. On the **Job Management** page, drag the **pgsql** icon on the left to the editing area on the right and double-click the icon to go to the PgSQL job configuration page. Configure parameters based on the following table. For details about job parameters, see **Creating a CDL Data Synchronization Job**.

   Table 2-47 PgSQL job parameters

Parameter	Example
Link	pgsqllink
Tasks Max	1
Mode	insert, update, and delete
dbName Alias	pgsqldb
Schema	pgschema
Slot Name	pg_slot
Enable FailOver Slot	No
Slot Drop	No
Connect With Hudi	Yes
Use Exist Publication	No
Publication Name	publicationtest

2. Click the plus sign (+) to display more parameters.



NOTE

- **Start Time**: indicates the start time of table synchronization.
- **WhiteList**: Enter a table in the database.
- **Blacklist**: Enter the database table that does not need to capture data.
- **Topic Table Mapping**
  - This parameter is mandatory if **Connect With Hudi** is set to **Yes**.
  - Enter the table name in the first text box, for example, **test**. Enter a topic name in the second text box, for example, **test_topic**. The topic name must match the table name in the first text box.

3. Click **OK**. The PgSQL job parameters are configured.

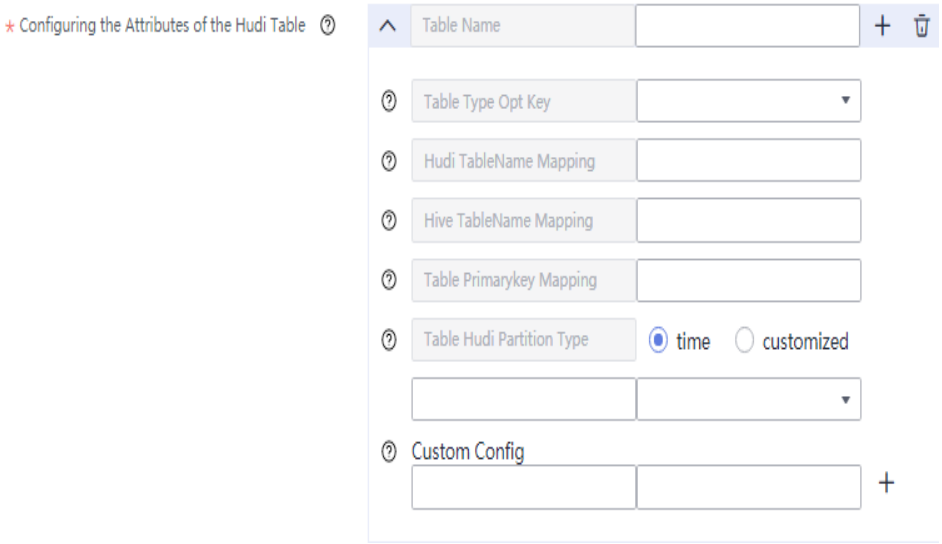
Step 7 Configure Hudi job parameters.

1. On the **Job Management** page, drag the **hudi** icon in the Sink area on the left to the editing area on the right and double-click the icon to go to the Hudi job configuration page. Configure parameters based on the following table. For details about job parameters, see **Creating a CDL Data Synchronization Job**.

Table 2-48 Sink Hudi job parameters

Parameter	Example Value
Link	hudilink
Path	/cdl/test
Interval	10
Max Rate Per Partition	0
Parallelism	10
Target Hive Database	default

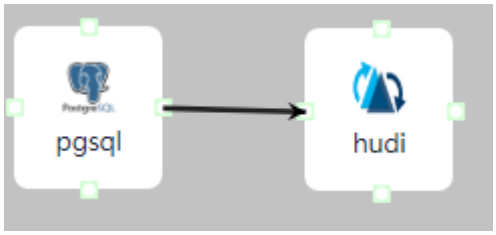
Parameter	Example Value
Configuring Hudi Table Attributes	Visual View
Global Configuration of Hudi Table Attributes	-
Configuring the Attributes of the Hudi Table: Table Name	test
Configuring the Attributes of the Hudi Table: Table Type Opt Key	COPY_ON_WRITE
Configuring the Attributes of the Hudi Table: Hudi TableName Mapping	-
Configuring the Attributes of the Hudi Table: Hive TableName Mapping	-
Configuring the Attributes of the Hudi Table: Table Primarykey Mapping	id
Configuring the Attributes of the Hudi Table: Table Hudi Partition Type	-
Configuring the Attributes of the Hudi Table: Custom Config	-



2. (Optional) Click the plus sign (+) to display the **Execution Env** parameter. Select a created environment for it. The default value is **defaultEnv**.

3. Click **OK**.

Step 8 Drag the two icons to associate the job parameters and click **Save**. The job configuration is complete.



Step 9 In the job list on the **Job Management** page, locate the created job, click **Start** in the **Operation** column, and wait until the job is started.

Check whether the data transmission takes effect, for example, insert data into the table in the PgSQL database and view the content of the file imported to Hudi.

----End

2.5.5 Synchronizing Data from openGauss to Hudi Using CDL

Scenario

Import data from OpenGauss to Hudi on the CDLService web UI of a cluster with Kerberos authentication enabled.

This section applies to MRS 3.3.0 and later versions.

Prerequisites

- The CDL and Hudi services have been installed in a cluster and are running properly.
- The write-ahead log function has been enabled for the OpenGauss database by referring to **Enabling the Write-Ahead Log Function for the OpenGauss Database**.
- You have created a human-machine user, for example, **cdluser**, added the user to user groups **cdladmin** (primary group), **hadoop**, **kafka**, and **supergroup**, and associated the user with the **System_administrator** role on FusionInsight Manager.

Procedure

Step 1 Log in to FusionInsight Manager as user **cdluser** (change the password upon your first login), choose **Cluster** > **Services** > **CDL**, and click the link next to **CDLService UI** to go to the CDLService web UI.

Step 2 Choose **Link Management** and click **Add Link**. On the displayed dialog box, set parameters for adding the **opengauss** and **hudi** links by referring to the following tables. **Creating a CDL Database Connection** describes the data link parameters.

Table 2-49 OpenGauss data link parameters

Parameter	Example
Link Type	opengauss
Name	opengausslink
Host	100.85.xxx.xxx
Port	8000
DB Name	opengaussdb
User	opengaussuser
Password	*opengauss user password*
Description	-

Table 2-50 Hudi data link parameters

Parameter	Example
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	xxx

Step 3 After the parameters are configured, click **Test** to check whether the data link is normal.

After the test is successful, click **OK**.

Step 4 (Optional) Choose **ENV Management** and click **Add Env**. In the displayed dialog box, configure the parameters based on the following table.

Table 2-51 Parameters for adding an ENV

Parameter	Example Value
Name	test-env
Driver Memory	1 GB
Type	spark
Executor Memory	1 GB

Parameter	Example Value
Executor Cores	1
Number Executors	1
Queue	-
Description	-

Click **OK**.

Step 5 Choose **Job Management** > **Data synchronization task** and click **Add Job**. In the displayed dialog box, set parameters. Click **Next**.

The following table shows the job parameters.

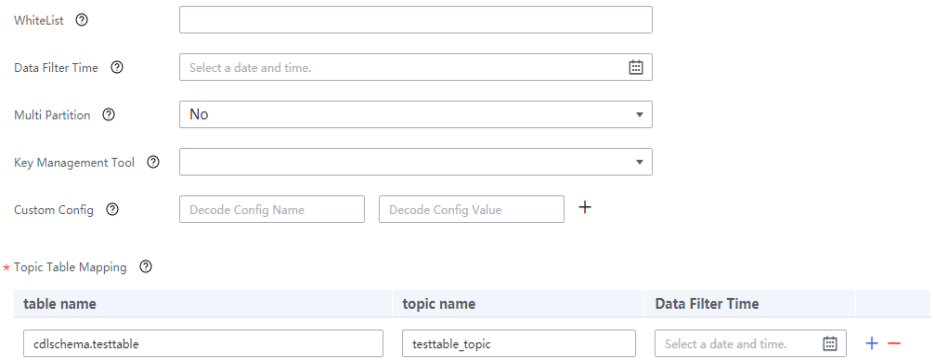
Parameter	Example
Name	job_opengaustohudi
Desc	-

Step 6 Configure OpenGauss job parameters.

1. On the **Job Management** page, drag the **opengauss** icon on the left to the editing area on the right and double-click the icon to go to the OpenGauss job configuration page. Set parameters by referring to the following table. **Creating a CDL Data Synchronization Job** describes the job parameters.

   Table 2-52 OpenGauss job parameters

Parameter	Example
Link	opengausslink
Tasks Max	1
Mode	**insert**, **update**, and **delete**
dbName Alias	opengaussdb
Slot Name	oct_twenty_two
Slot Drop	No
Connect With Hudi	Yes
Topic Table Mapping	cdlschema.testtable/testtable_topic



2. Click **OK**. The OpenGauss job parameters are configured.

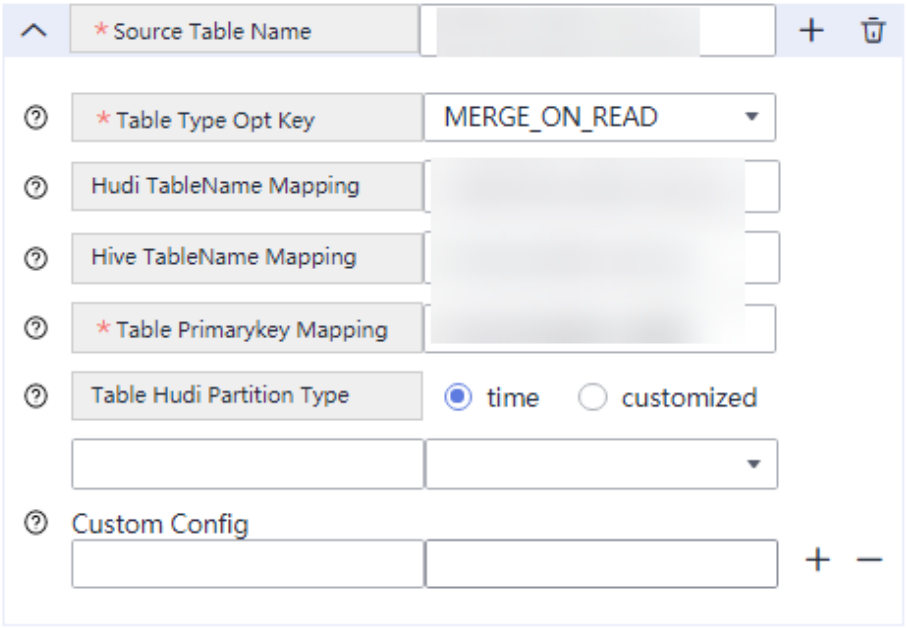
Step 7 Configure Hudi job parameters.

1. On the **Job Management** page, drag the **hudi** icon in the Sink area on the left to the editing area on the right and double-click the icon to go to the Hudi job configuration page. Set parameters by referring to the following table. **Creating a CDL Data Synchronization Job** describes the job parameters.

Table 2-53 Sink Hudi job parameters

Parameter	Example
Link	hudilink
Path	/cdl/test
Interval	5
Max Rate Per Partition	0
Parallelism	10
Configuring Hudi Table Attributes	Visual View
Global Configuration of Hudi Table Attributes	-
Configuring the Attributes of the Hudi Table: Source Table Name	cdlschema.testtable
Configuring the Attributes of the Hudi Table: Table Type Opt Key	MERGE_ON_READ
Configuring the Attributes of the Hudi Table: Hudi TableName Mapping	**testtable** or **/cdlschema/testtable**
Configuring the Attributes of the Hudi Table: Hive TableName Mapping	cdlschema.testtable

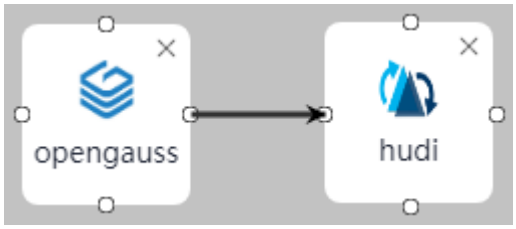
Parameter	Example
Configuring the Attributes of the Hudi Table: Table Primarykey Mapping	so_line_id,order_number
Configuring the Attributes of the Hudi Table: Table Hudi Partition Type	time
Configuring the Attributes of the Hudi Table: Custom Config	-



2. (Optional) Select the created ENV. The default value is **defaultEnv**.

3. Click **OK**.

Step 8 Drag the two icons to associate the job parameters and click **Save**. The job configuration is complete.



Step 9 In the job list on the **Job Management** page, locate the created job, click **Start** in the **Operation** column, and wait until the job is started.

Check whether the data transmission takes effect, for example, insert data into the table in the OpenGauss database and view the content of the file imported to Hudi.

----End

2.5.6 Synchronizing Data from Hudi to DWS Using CDL

Scenario

This section describes how to import data from Hudi to DWS by using the CDLService web UI of a cluster with Kerberos authentication enabled in MRS 3.2.0.

Prerequisites

- The CDL and Hudi services have been installed in a cluster and are running properly.
- The prerequisites for the DWS database have been met. For details, see **Prerequisites for the DWS database**.
- You have created a human-machine user, for example, **cdluser**, added the user to user groups **cdladmin** (primary group), **hadoop**, **kafka**, and **supergroup**, and associated the user with the **System_administrator** role on FusionInsight Manager.

Procedure

Step 1 Log in to FusionInsight Manager as user **cdluser** (change the password upon your first login), choose **Cluster** > **Services** > **CDL**, and click the link next to **CDLService UI** to go to the CDLService web UI.

Step 2 Choose **Link Management** and click **Add Link**. In the displayed dialog box, set parameters for adding the **dws** and **hudi** links by referring to the following tables. **Creating a CDL Database Connection** describes the data link parameters.

Table 2-54 DWS data link parameters

Parameter	Example
Link Type	dws
Name	dwstest
Host	10.10.10.10
Port	8000
DB Name	dwsdb
User	dbuser
Password	*Password of the **dbuser** user*
Description	-

Table 2-55 Hudi data link parameters

Parameter	Example
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	xxx

Step 3 After the parameters are configured, click **Test** to check whether the data link is normal.

After the test is successful, click **OK**.

Step 4 (Optional) Choose **ENV Management** and click **Add Env**. In the displayed dialog box, configure the parameters based on the following table.

Table 2-56 Parameters for adding an ENV

Parameter	Example Value
Name	test-env
Driver Memory	1 GB
Type	spark
Executor Memory	1 GB
Executor Cores	1
Number Executors	1
Queue	-
Description	-

Click **OK**.

Step 5 Choose **Job Management** > **Data synchronization task** and click **Add Job**. In the displayed dialog box, set parameters and click **Next**.

Parameters are as follows.

Parameter	Example
Name	job_huditodws

Parameter	Example
Desc	-

Step 6 Configure Hudi job parameters.

1. On the **Job Management** page, drag the **hudi** icon in the Source area on the left to the editing area on the right and double-click the icon to go to the Hudi job configuration page. Configure parameters based on the following table. For details about job parameters, see **Creating a CDL Data Synchronization Job**.

   Table 2-57 Source Hudi job parameters

Parameter	Example
Link	hudilink
Interval	10
Table Info	{"table1":[{"source.database":"dwsdb","source.tablename":"tabletest","target.tablename":"tabletest"}]}

2. Click **OK**. The Hudi job parameters are configured.

Step 7 Configure DWS job parameters.

1. On the **Job Management** page, drag the **dws** icon on the left to the editing area on the right and double-click the icon to go to the DWS job configuration page. Configure parameters based on the following table. For details about job parameters, see **Creating a CDL Data Synchronization Job**.

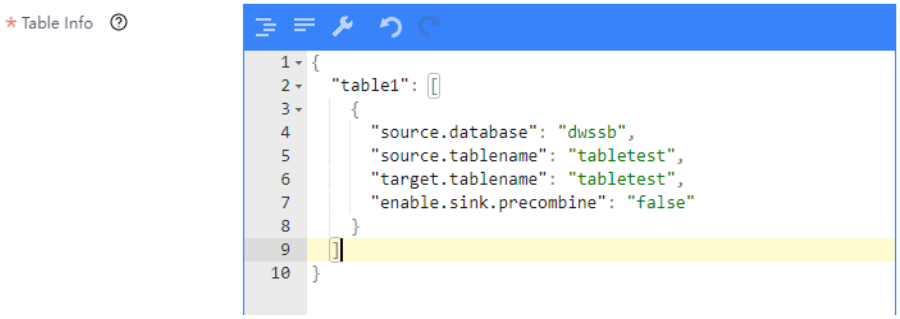
   Table 2-58 DWS job parameters

Parameter	Example
Link	dwstest

Parameter	Example
Query Timeout	180000
Batch Size	10

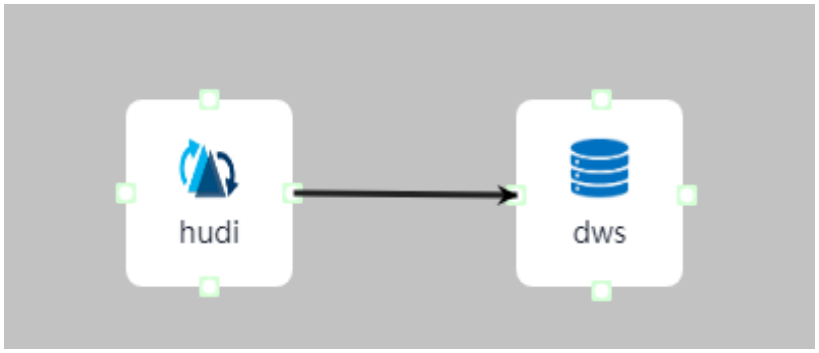
NOTE

- Data can be synchronized from Hudi to GaussDB(DWS) only when both Hudi and GaussDB(DWS) contain the **precombine** field.
- GaussDB(DWS) tables must contain the **precombine** field and the primary key.
- By default, the Hudi built-in field **_hoodie_event_time** is used. If this field is not used, **enable.sink.precombine** must be specified. An example is as follows:



2. Click **OK**.

Step 8 Drag the two icons to associate the job parameters and click **Save**. The job configuration is complete.



Step 9 In the job list on the **Job Management** page, locate the created job, click **Start** in the **Operation** column, and wait until the job is started.

Check whether the data transmission takes effect, for example, insert data into the Hudi table and view the content of the file imported to DWS.

----End

2.5.7 Synchronizing Data from Hudi to ClickHouse Using CDL

Scenario

This section describes how to import data from Hudi to ClickHouse by using the CDLService web UI of a cluster with Kerberos authentication enabled in MRS 3.2.0.

Prerequisites

- The CDL, Hudi, and ClickHouse services have been installed in a cluster and are running properly.
- You have operation permissions on ClickHouse. For details, see **Creating a ClickHouse Role**.
- You have created a human-machine user with the ClickHouse administrator permissions (for details, see **Creating a ClickHouse Role**), for example, **cdluser**, added the user to user groups **cdladmin** (primary group), **hadoop**, **kafka**, and **supergroup**, and associated the user with the **System_administrator** role on FusionInsight Manager.
- You have manually created a local table and distributed table on ClickHouse. The local table uses the ReplicatedReplacingMergeTree engine. For details, see **ClickHouse Client Practices**.

Procedure

Step 1 Log in to FusionInsight Manager as user **cdluser** (change the password upon your first login), choose **Cluster** > **Services** > **CDL**, and click the link next to **CDLService UI** to go to the CDLService web UI.

Step 2 Choose **Link Management** and click **Add Link**. In the displayed dialog box, set parameters for adding the **clickhouse** and **hudi** links by referring to the following tables. **Creating a CDL Database Connection** describes the data link parameters.

Table 2-59 ClickHouse data link parameters

Parameter	Example
Link Type	clickhouse
Name	cklink
Host	10.10.10.10:21428
User	*cdluser*
Password	*Password of the **cdluser** user*
Description	-

Table 2-60 Hudi data link parameters

Parameter	Example
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	-

Step 3 After the parameters are configured, click **Test** to check whether the data link is normal.

After the test is successful, click **OK**.

Step 4 (Optional) Choose **ENV Management** and click **Add Env**. In the displayed dialog box, configure the parameters based on the following table.

Table 2-61 Parameters for adding an ENV

Parameter	Example Value
Name	test-env
Driver Memory	1 GB
Type	spark
Executor Memory	1 GB
Executor Cores	1
Number Executors	1
Queue	-
Description	-

Click **OK**.

Step 5 Choose **Job Management** > **Data synchronization task** and click **Add Job**. In the displayed dialog box, set parameters and click **Next**.

Parameters are as follows.

Parameter	Example
Name	job_huditock

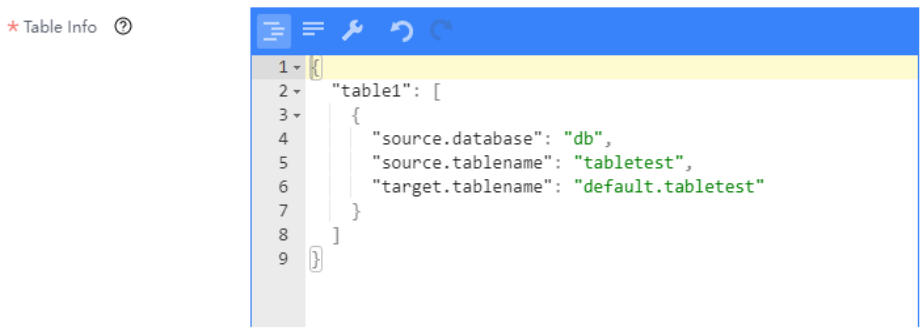
Parameter	Example
Desc	-

Step 6 Configure Hudi job parameters.

1. On the **Job Management** page, drag the **hudi** icon in the Source area on the left to the editing area on the right and double-click the icon to go to the Hudi job configuration page. Configure parameters based on the following table. For details about job parameters, see **Creating a CDL Data Synchronization Job**.

   Table 2-62 Source Hudi job parameters

Parameter	Example
Link	hudilink
Interval	10
Table Info	{"table1":[{"source.database":"db","source.tablename":"tabletest","target.tablename":"default.tabletest"}]} **NOTE** You do not need to configure the fields provided by Hudi. You only need to configure the service fields that need to be synchronized to ClickHouse.



2. Click **OK**. The Hudi job parameters are configured.

Step 7 Configure ClickHouse job parameters.

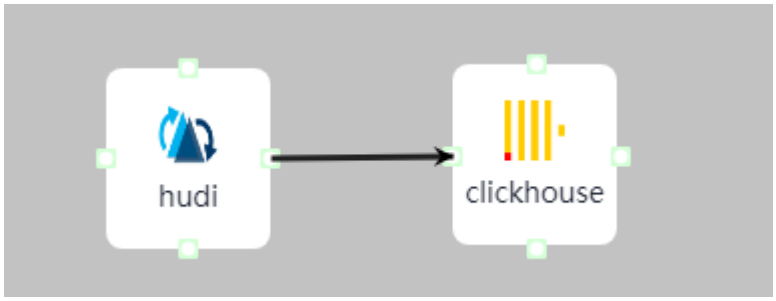
1. On the **Job Management** page, drag the **clickhouse** icon on the left to the editing area on the right and double-click the icon to go to the ClickHouse job configuration page. Configure parameters based on the following table. For details about job parameters, see **Creating a CDL Data Synchronization Job**.

Table 2-63 ClickHouse job parameters

Parameter	Example
Link	cklink
Query Timeout	60000
Batch Size	100

2. Click **OK**.

Step 8 Drag the two icons to associate the job parameters and click **Save**. The job configuration is complete.



Step 9 In the job list on the **Job Management** page, locate the created job, click **Start** in the **Operation** column, and wait until the job is started.

Check whether the data transmission takes effect, for example, insert data into the Hudi table and view the content of the file imported to ClickHouse.

----End

2.5.8 Synchronizing openGauss Data to Hudi Using CDL (ThirdKafka)

Scenario

This section describes how to import openGauss data from ThirdKafka to Hudi by using the CDLService web UI of a cluster with Kerberos authentication enabled in MRS 3.2.0.

Prerequisites

- The CDL and Hudi services have been installed in a cluster and are running properly.

- Topics of the ThirdKafka database can be consumed by the MRS cluster. For details, see **Prerequisites for ThirdPartyKafka**.
- You have created a human-machine user, for example, **cdluser**, added the user to user groups **cdladmin** (primary group), **hadoop**, **kafka**, and **supergroup**, and associated the user with the **System_administrator** role on FusionInsight Manager.

Procedure

Step 1 Log in to FusionInsight Manager as user **cdluser** (change the password upon your first login), choose **Cluster** > **Services** > **CDL**, and click the link next to **CDLService UI** to go to the CDLService web UI.

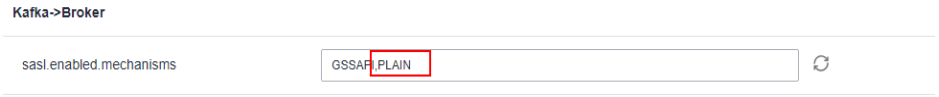
Step 2 Choose **Link Management** and click **Add Link**. In the displayed dialog box, set parameters for adding the **thirdparty-kafka** and **hudi** links by referring to the following tables. **Creating a CDL Database Connection** describes the data link parameters.

Table 2-64 thirdparty-kafka data link parameters

Parameter	Example
Name	opengausslink
Link Type	thirdparty-kafka
Bootstrap Servers	10.10.10.10:9093
Security Protocol	SASL_SSL
Username	testuser
Password	*Password of the **testuser** user*
SSL Truststore Location	Click **Upload** to upload the authentication file.
SSL Truststore Password	-
Datastore Type	opengauss
Host	11.11.xxx.xxx,12.12.xxx.xxx
Port	8000
DB Name	opengaussdb
User	opengaussuser
DB Password	*Password of the **opengaussuser** user*
Description	-

NOTE

MRS Kafka can also be used as the source of thirdparty-kafka. If the username and password are used for login authentication, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, click **Configuration**, search for the **sasl.enabled.mechanisms** parameter in the search box, add **PLAIN** as the parameter value, click **Save** to save the configuration, and restart the Kafka service for the configuration to take effect.



On the CDL web UI, configure the thirdparty-kafka link that uses MRS Kafka as the source. For example, the data link configuration is as follows:

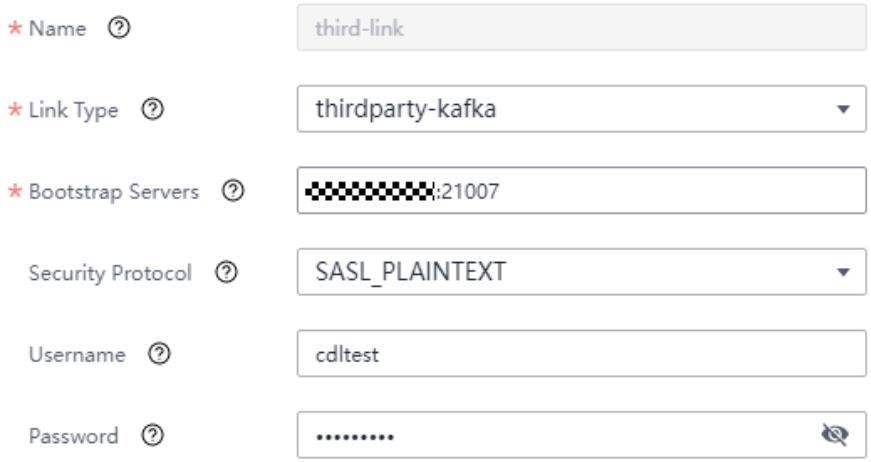


Table 2-65 Hudi data link parameters

Parameter	Example
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	-

Step 3 After the parameters are configured, click **Test** to check whether the data link is normal.

After the test is successful, click **OK**.

Step 4 (Optional) Choose **ENV Management** and click **Add Env**. In the displayed dialog box, configure the parameters based on the following table.

Table 2-66 Parameters for adding an ENV

Parameter	Example Value
Name	test-env
Driver Memory	1 GB
Type	spark
Executor Memory	1 GB
Executor Cores	1
Number Executors	1
Queue	-
Description	-

Click **OK**.

Step 5 Choose **Job Management** > **Data synchronization task** and click **Add Job**. In the displayed dialog box, set parameters and click **Next**.

Parameters are as follows.

Parameter	Example
Name	job_opengausstohudi
Desc	New CDL Job

Step 6 Configure ThirdKafka job parameters.

1. On the **Job Management** page, drag the **thirdparty-kafka** icon on the left to the editing area on the right and double-click the icon to go to the ThirdpartyKafka job configuration page. Configure parameters based on the following table. For details about job parameters, see **Creating a CDL Data Synchronization Job**.

   Table 2-67 thirdparty-kafka job parameters

Parameter	Example
Link	opengausslink
DB Name	opengaussdb
Schema	opengausssschema
Datastore Type	opengauss
Source Topics	source_topic
Tasks Max	1

Parameter	Example
Tolerance	none
Start Time	-
Multi Partition	No
Topic Table Mapping	test/hudi_topic

2. Click **OK**. The ThirdpartyKafka job parameters are configured.

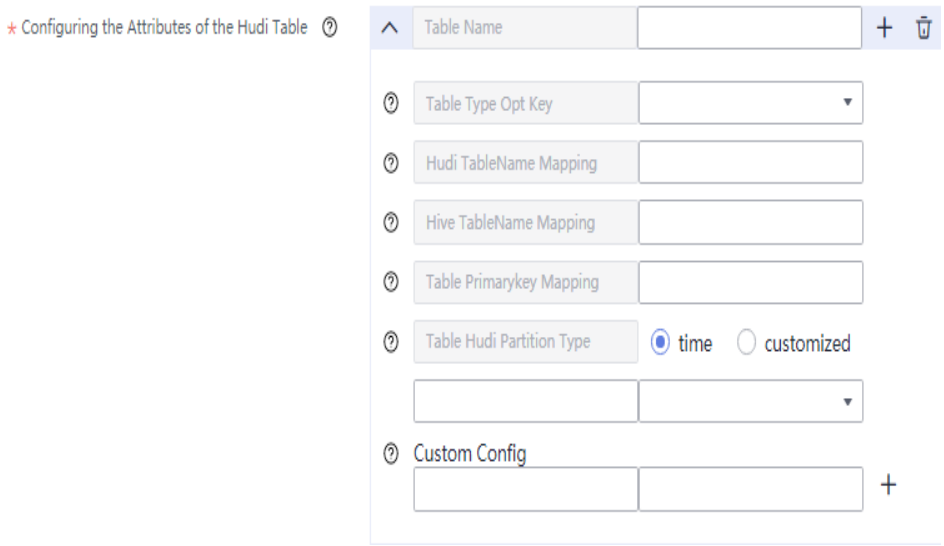
Step 7 Configure Hudi job parameters.

1. On the **Job Management** page, drag the **hudi** icon in the Sink area on the left to the editing area on the right and double-click the icon to go to the Hudi job configuration page. Configure parameters based on the following table. For details about job parameters, see **Creating a CDL Data Synchronization Job**.

Table 2-68 Sink Hudi job parameters

Parameter	Example Value
Link	hudilink
Path	/cdl/test
Interval	10
Max Rate Per Partition	0
Parallelism	10
Target Hive Database	default
Configuring Hudi Table Attributes	Visual View
Global Configuration of Hudi Table Attributes	-

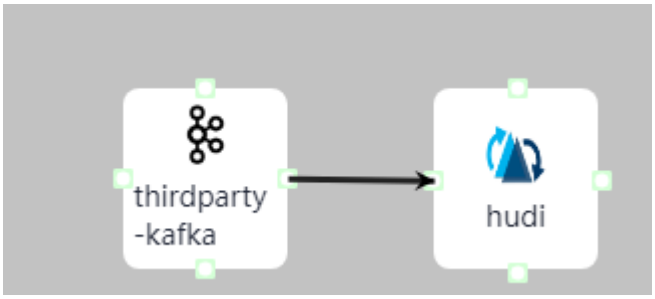
Parameter	Example Value
Configuring the Attributes of the Hudi Table: Table Name	test
Configuring the Attributes of the Hudi Table: Table Type Opt Key	COPY_ON_WRITE
Configuring the Attributes of the Hudi Table: Hudi TableName Mapping	-
Configuring the Attributes of the Hudi Table: Hive TableName Mapping	-
Configuring the Attributes of the Hudi Table: Table Primarykey Mapping	id
Configuring the Attributes of the Hudi Table: Table Hudi Partition Type	-
Configuring the Attributes of the Hudi Table: Custom Config	-



2. (Optional) Click the plus sign (+) to display the **Execution Env** parameter. Select a created environment for it. The default value is **defaultEnv**.

3. Click **OK**.

Step 8 Drag the two icons to associate the job parameters and click **Save**. The job configuration is complete.



Step 9 In the job list on the **Job Management** page, locate the created job, click **Start** in the **Operation** column, and wait until the job is started.

Check whether the data transmission takes effect, for example, insert data into the table in the openGauss database and view the content of the file imported to Hudi.

----End

2.5.9 Synchronizing drs-oracle-json Database to Hudi Using CDL (ThirdKafka)

Scenario

Import Oracle database data from ThirdKafka to Hudi on the CDLService web UI of a cluster with Kerberos authentication enabled.

This section applies to MRS 3.3.0 or later.

Prerequisites

- The CDL and Hudi services have been installed in a cluster and are running properly.
- Topics of the ThirdKafka database can be consumed by the MRS cluster. For details, see **Prerequisites for ThirdPartyKafka**.
- You have created a human-machine user, for example, **cdluser**, added the user to user groups **cdladmin** (primary group), **hadoop**, **kafka**, and **supergroup**, and associated the user with the **System_administrator** role on FusionInsight Manager.

Procedure

Step 1 Log in to FusionInsight Manager as user **cdluser** (change the password upon your first login), choose **Cluster** > **Services** > **CDL**, and click the link next to **CDLService UI** to go to the CDLService web UI.

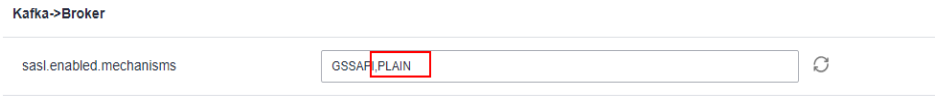
Step 2 Choose **Link Management** and click **Add Link**. On the displayed dialog box, set parameters for adding the **thirdparty-kafka** and **hudi** links by referring to the following tables. **Creating a CDL Database Connection** describes the data link parameters.

Table 2-69 thirdparty-kafka data link parameters

Parameter	Example Value
Name	oraclelink
Link Type	thirdparty-kafka
Bootstrap Servers	10.10.10.10:9093
Security Protocol	SASL_SSL
Username	testuser
Password	*Password of the **testuser** user*
SSL Truststore Location	Click **Upload** to upload the authentication file.
SSL Truststore Password	-
Datastore Type	drs-oracle-json
Description	thirdparty-kafka Link

NOTE

MRS Kafka can also be used as the source of thirdparty-kafka. If the username and password are used for login authentication, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, click **Configurations**, search for the **sasl.enabled.mechanisms** parameter in the search box, add **PLAIN** as the parameter value, click **Save** to save the configuration, and restart the Kafka service for the configuration to take effect.



On the CDL web UI, configure the thirdparty-kafka link that uses MRS Kafka as the source. For example, the data link configuration is as follows:

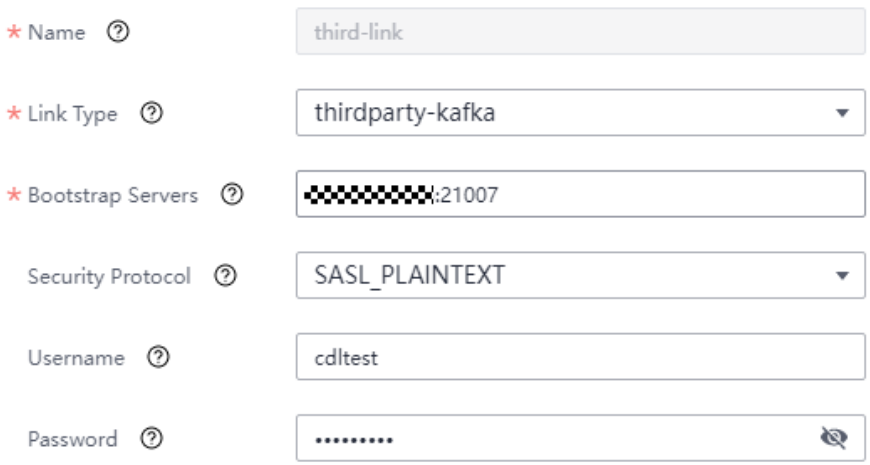


Table 2-70 Hudi data link parameters

Parameter	Example Value
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	-

Step 3 After the parameters are configured, click **Test** to check whether the data link is normal.

After the test is successful, click **OK**.

Step 4 (Optional) Choose **ENV Management** and click **Add Env**. In the displayed dialog box, configure the parameters based on the following table.

Table 2-71 Parameters for adding an ENV

Parameter	Example Value
Name	test-env
Driver Memory	1 GB
Type	spark
Executor Memory	1 GB
Executor Cores	1
Number Executors	1
Queue	-
Description	-

Click **OK**.

Step 5 Choose **Job Management** > **Data synchronization task** and click **Add Job**. In the displayed dialog box, set parameters. Click **Next**.

Specifically:

Parameter	Example Value
Name	job_oracletohudi

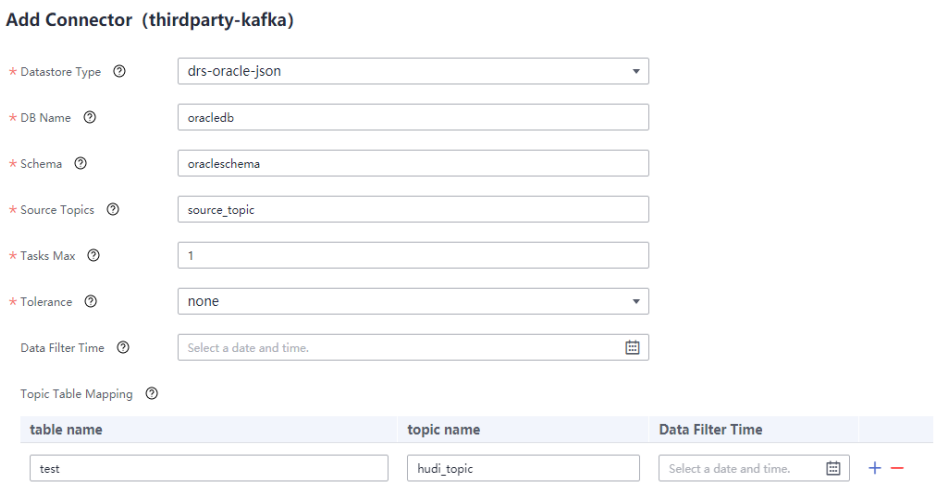
Parameter	Example Value
Desc	New CDL Job

Step 6 Configure ThirdKafka job parameters.

1. On the **Job Management** page, drag the **thirdparty-kafka** icon on the left to the editing area on the right and double-click the icon to go to the ThirdpartyKafka job configuration page. Set parameters by referring to the following table. **Creating a CDL Data Synchronization Job** describes the job parameters.

   Table 2-72 thirdparty-kafka job parameters

Parameter	Example Value
Link	oraclelink
DB Name	oracledb
Schema	oracleschema
Datastore Type	drs-oracle-json
Source Topics	source_topic
Tasks Max	1
Tolerance	none
Data Filter Time	-
Topic Table Mapping	test/hudi_topic



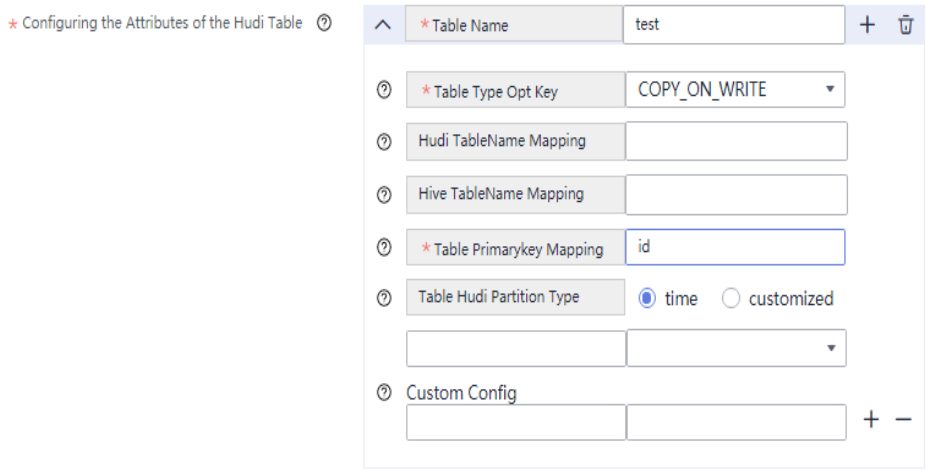
2. Click **OK**. The ThirdpartyKafka job parameters are configured.

Step 7 Configure Hudi job parameters.

1. On the **Job Management** page, drag the **hudi** icon in the Sink area on the left to the editing area on the right and double-click the icon to go to the Hudi job configuration page. Set parameters by referring to the following table. **Creating a CDL Data Synchronization Job** describes the job parameters.

   Table 2-73 Sink Hudi job parameters

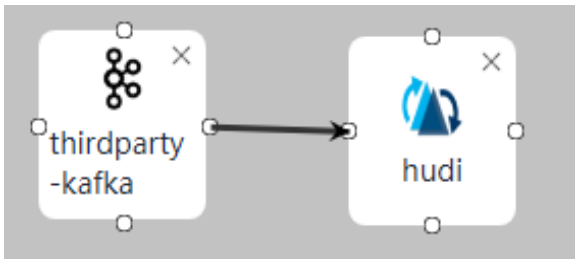
Parameter	Example Value
Link	hudilink
Path	/cdl/test
Interval	10
Max Rate Per Partition	0
Parallelism	10
Target Hive Database	default
Configuring Hudi Table Attributes	Visual View
Global Configuration of Hudi Table Attributes	-
Configuring the Attributes of the Hudi Table: Table Name	test
Configuring the Attributes of the Hudi Table: Table Type Opt Key	COPY_ON_WRITE
Configuring the Attributes of the Hudi Table: Hudi TableName Mapping	-
Configuring the Attributes of the Hudi Table: Hive TableName Mapping	-
Configuring the Attributes of the Hudi Table: Table Primarykey Mapping	id
Configuring the Attributes of the Hudi Table: Table Hudi Partition Type	-
Configuring the Attributes of the Hudi Table: Custom Config	-



2. (Optional) Click the plus sign (+) to display the **Execution Env** parameter. Select a created environment for it. The default value is **defaultEnv**.

3. Click **OK**.

Step 8 Drag the two icons to associate the job parameters and click **Save**. The job configuration is complete.



Step 9 In the job list on the **Job Management** page, locate the created job, click **Start** in the **Operation** column, and wait until the job is started.

Check whether the data transmission takes effect, for example, insert data into the table in the Oracle database and view the content of the file imported to Hudi.

----End

2.5.10 Synchronizing drs-oracle-avro Database to Hudi Using CDL (ThirdKafka)

Scenario

Import drs-avro-oracle database data from ThirdKafka to Hudi on the CDLService web UI of a cluster with Kerberos authentication enabled.

This section applies to MRS 3.3.0 or later.

Prerequisites

- The CDL and Hudi services have been installed in a cluster and are running properly.
- Topics of the ThirdKafka database can be consumed by the MRS cluster. For details, see **Prerequisites for ThirdPartyKafka**.
- You have created a human-machine user, for example, **cdluser**, added the user to user groups **cdladmin** (primary group), **hadoop**, **kafka**, and **supergroup**, and associated the user with the **System_administrator** role on FusionInsight Manager.

Procedure

Step 1 Log in to FusionInsight Manager as user **cdluser** (change the password upon your first login), choose **Cluster** > **Services** > **CDL**, and click the link next to **CDLService UI** to go to the CDLService web UI.

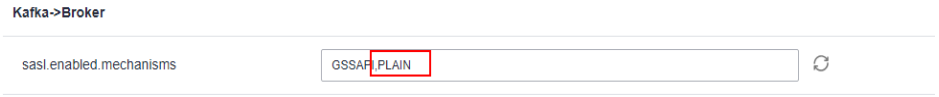
Step 2 Choose **Link Management** and click **Add Link**. On the displayed dialog box, set parameters for adding the **thirdparty-kafka** and **hudi** links by referring to the following tables. **Creating a CDL Database Connection** describes the data link parameters.

Table 2-74 thirdparty-kafka data link parameters

Parameter	Example Value
Name	drs_avro_oracle_link
Link Type	thirdparty-kafka
Bootstrap Servers	10.10.10.10:9093
Security Protocol	SASL_SSL
Username	testuser
Password	*Password of the **testuser** user*
SSL Truststore Location	Click **Upload** to upload the authentication file.
SSL Truststore Password	-
Datastore Type	drs-oracle-avro
Description	thirdparty-kafka Link

NOTE

MRS Kafka can also be used as the source of thirdparty-kafka. If the username and password are used for login authentication, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, click **Configurations**, search for the **sasl.enabled.mechanisms** parameter in the search box, add **PLAIN** as the parameter value, click **Save** to save the configuration, and restart the Kafka service for the configuration to take effect.



On the CDL web UI, configure the thirdparty-kafka link that uses MRS Kafka as the source. For example, the data link configuration is as follows:

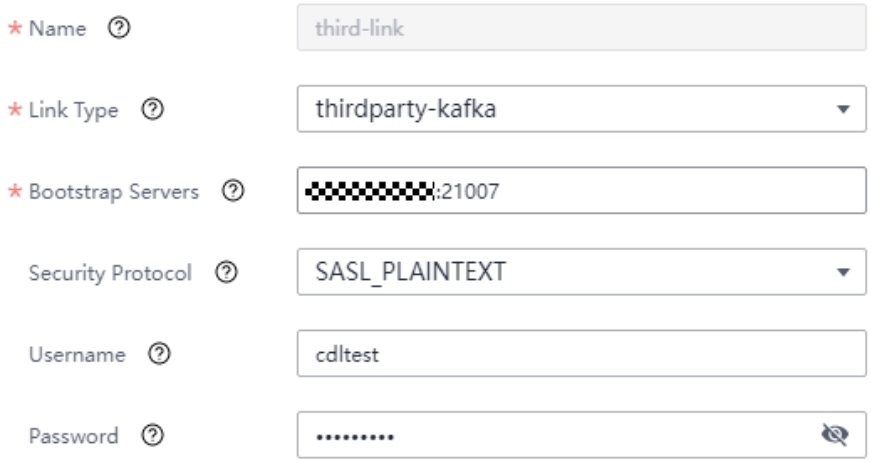


Table 2-75 Hudi data link parameters

Parameter	Example Value
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	-

Step 3 After the parameters are configured, click **Test** to check whether the data link is normal.

After the test is successful, click **OK**.

Step 4 (Optional) Choose **ENV Management** and click **Add Env**. In the displayed dialog box, configure the parameters based on the following table.

Table 2-76 Parameters for adding an ENV

Parameter	Example Value
Name	test-env
Driver Memory	1 GB
Type	spark
Executor Memory	1 GB
Executor Cores	1
Number Executors	1
Queue	-
Description	-

Click **OK**.

Step 5 Choose **Job Management** > **Data synchronization task** and click **Add Job**. In the displayed dialog box, set parameters. Click **Next**.

Specifically:

Parameter	Example Value
Name	job_avro_oracletohudi
Desc	New CDL Job

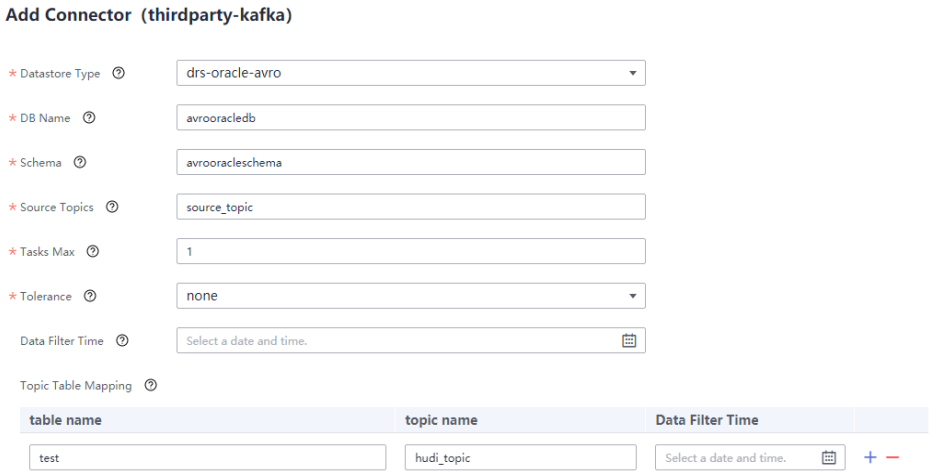
Step 6 Configure ThirdKafka job parameters.

1. On the **Job Management** page, drag the **thirdparty-kafka** icon on the left to the editing area on the right and double-click the icon to go to the ThirdpartyKafka job configuration page. Set parameters by referring to the following table. **Creating a CDL Data Synchronization Job** describes the job parameters.

   Table 2-77 thirdparty-kafka job parameters

Parameter	Example Value
Link	job_avro_oracletohudi
DB Name	avrooracledb
Schema	avrooracleschema
Datastore Type	drs-oracle-avro
Source Topics	source_topic
Tasks Max	1

Parameter	Example Value
Tolerance	none
Data Filter Time	-
Topic Table Mapping	test/hudi_topic



2. Click **OK**. The ThirdpartyKafka job parameters are configured.

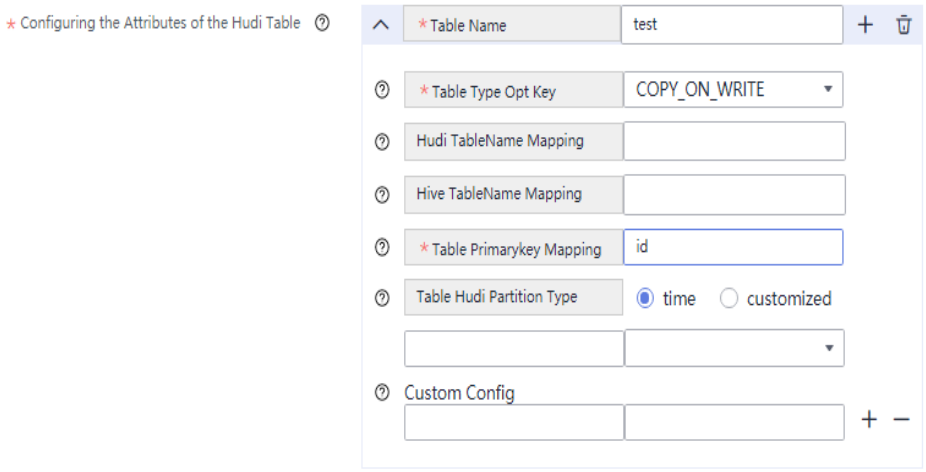
Step 7 Configure Hudi job parameters.

1. On the **Job Management** page, drag the **hudi** icon in the Sink area on the left to the editing area on the right and double-click the icon to go to the Hudi job configuration page. Set parameters by referring to the following table. **Creating a CDL Data Synchronization Job** describes the job parameters.

Table 2-78 Sink Hudi job parameters

Parameter	Example Value
Link	hudilink
Path	/cdl/test
Interval	10
Max Rate Per Partition	0
Parallelism	10
Target Hive Database	default
Configuring Hudi Table Attributes	Visual View
Global Configuration of Hudi Table Attributes	-

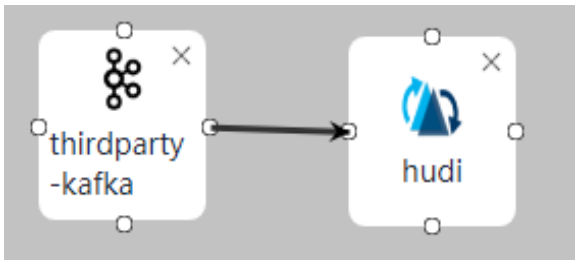
Parameter	Example Value
Configuring the Attributes of the Hudi Table: Table Name	test
Configuring the Attributes of the Hudi Table: Table Type Opt Key	COPY_ON_WRITE
Configuring the Attributes of the Hudi Table: Hudi TableName Mapping	-
Configuring the Attributes of the Hudi Table: Hive TableName Mapping	-
Configuring the Attributes of the Hudi Table: Table Primarykey Mapping	id
Configuring the Attributes of the Hudi Table: Table Hudi Partition Type	-
Configuring the Attributes of the Hudi Table: Custom Config	-



2. (Optional) Click the plus sign (+) to display the **Execution Env** parameter. Select a created environment for it. The default value is **defaultEnv**.

3. Click **OK**.

Step 8 Drag the two icons to associate the job parameters and click **Save**. The job configuration is complete.



Step 9 In the job list on the **Job Management** page, locate the created job, click **Start** in the **Operation** column, and wait until the job is started.

Check whether the data transmission takes effect, for example, insert data into the table in the drs-avro-oracle database and view the content of the file imported to Hudi.

----End

2.6 CDL Job DDL Changes

DDL operations include creating databases or tables, changing table field types and table field names, and adding or deleting table columns. Currently, MRS Change Data Loader (CDL) supports only DDL operations for synchronizing data from PostgreSQL to Hudi. The sequence of DDL operations is as follows:

1. Stop a CDL job.
2. Execute DDL operations on the Hudi side.
3. Perform DDL operations on the source database.

This section applies to MRS 3.3.0 and later versions. It provides guidance on DDL operations, such as adding fields, changing field types, changing field names, and deleting fields.

Adding a Field

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. On the displayed page, click the hyperlink on the right of **CDLService UI** to go to the CDLService web UI. On the data synchronization job list page, locate the row that contains the target DDL change job, and choose **More** > **Stop** to stop a CDL job.

Step 2 Log in to the node where the client is installed as the client installation user and run the following commands:

cd *Client installation directory*

source bigdata_env

source Hudi/component_env

kinit *Component service user* (Skip this step if Kerberos authentication is disabled for the cluster.)

Step 3 Run the following commands to log in to the Spark SQL CLI:

cd Spark/spark/bin

./spark-sql

Step 4 Run the following command:

set hoodie.schema.evolution.enable=true;

Step 5 Run the following command to add a field to the table:

alter table *tableName* **add columns(***columnName columnType***);**

Step 6 Add the column name and data type same as those in Hudi to the source database.

Step 7 Start the CDL task stopped in **Step 1** on the CDL web UI.

----End

Modifying a Field

NOTE

During field type conversion, ensure that the data type of the source value can be correctly converted to the target type. If the data type is incompatible, the conversion may fail. As a result, the task fails.

- Change the data type from VARCHAR to NUMBER.
  a. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. On the displayed page, click the hyperlink on the right of **CDLService UI** to go to the CDLService web UI. On the data synchronization job list page, locate the row that contains the target DDL change job, and choose **More** > **Stop** to stop a CDL job.
  b. Log in to the node where the client is installed as the client installation user and run the following commands:

     cd *Client installation directory*

     source bigdata_env

     source Hudi/component_env

     kinit *Component service user* (Skip this step if Kerberos authentication is disabled for the cluster.)
  c. Run the following commands to log in to the Spark SQL CLI:

     cd Spark/spark/bin

     ./spark-sql
  d. Run the following statement to change the data type from **string** to **decimal** on Hudi:

     ALTER TABLE *ddltest* **ALTER COLUMN string TYPE decimal(20,10);**
  e. Insert data into the source database. The data can be properly written to Hudi.
  f. In the source database, change the data type from **VARCHAR** to **NUMBER**.
  g. On the CDL web UI, start a task to update data in the source database.
- Change the data type from **NUMBER** to **VARCHAR**.
  a. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. On the displayed page, click the hyperlink on the right of **CDLService UI** to go to the CDLService web UI. On the data synchronization job list

page, locate the row that contains the target DDL change job, and choose **More** > **Stop** to stop a CDL job.

b. Log in to the node where the client is installed as the client installation user and run the following commands:

cd *Client installation directory*

source bigdata_env

source Hudi/component_env

kinit *Component service user* (Skip this step if Kerberos authentication is disabled for the cluster.)

c. Run the following commands to log in to the Spark SQL CLI:

cd Spark/spark/bin

./spark-sql

d. Run the following statement to change the data type from **decimal** to **string** on Hudi:

ALTER TABLE *ddltest* **ALTER COLUMN decimal TYPE string;**

e. Insert data into the source database. The data can be properly written to Hudi.

f. In the source database, change the data type from **NUMBER** to **VARCHAR**.

g. On the CDL web UI, start a task to update data in the source database.

- Change the data type from **DATE** to **VARCHAR**.

a. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. On the displayed page, click the hyperlink on the right of **CDLService UI** to go to the CDLService web UI. On the data synchronization job list page, locate the row that contains the target DDL change job, and choose **More** > **Stop** to stop a CDL job.

b. Log in to the node where the client is installed as the client installation user and run the following commands:

cd *Client installation directory*

source bigdata_env

source Hudi/component_env

kinit *Component service user* (Skip this step if Kerberos authentication is disabled for the cluster.)

c. Run the following commands to log in to the Spark SQL CLI:

cd Spark/spark/bin

./spark-sql

d. Run the following statement to change the data type from **date** to **string** on Hudi:

ALTER TABLE *ddltest2* **ALTER COLUMN date TYPE string;**

e. Insert data into the source database. The data can be properly written to Hudi.

f. In the source database, change the data type from **DATE** to **VARCHAR**.

g. On the CDL web UI, start a task to update data in the source database.

- Change the **DATA** type without a time zone to the **DATA** type with a time zone.
  a. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. On the displayed page, click the hyperlink on the right of **CDLService UI** to go to the CDLService web UI. On the data synchronization job list page, locate the row that contains the target DDL change job, and choose **More** > **Stop** to stop a CDL job.
  b. In the source database, change the data type from **timestamp** to **timestamptz**.
  c. Insert data into the source database. The data can be properly written to Hudi.
  d. On the CDL web UI, start a task to update data in the source database.
- Extend strings.
  a. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. On the displayed page, click the hyperlink on the right of **CDLService UI** to go to the CDLService web UI. On the data synchronization job list page, locate the row that contains the target DDL change job, and choose **More** > **Stop** to stop a CDL job.
  b. Extend the string length in the source database.
  c. Insert data into the source database. The data can be properly written to Hudi.
  d. On the CDL web UI, start a task to update data in the source database.
- Increase the decimal precision.
  a. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. On the displayed page, click the hyperlink on the right of **CDLService UI** to go to the CDLService web UI. On the data synchronization job list page, locate the row that contains the target DDL change job, and choose **More** > **Stop** to stop a CDL job.
  b. Log in to the node where the client is installed as the client installation user and run the following commands:

     cd *Client installation directory*

     source bigdata_env

     source Hudi/component_env

     kinit *Component service user* (Skip this step if Kerberos authentication is disabled for the cluster.)
  c. Run the following commands to log in to the Spark SQL CLI:

     cd Spark/spark/bin

     ./spark-sql
  d. Run the following statement to increase the decimal precision on Hudi:

     ALTER TABLE *ddltest2* **ALTER COLUMN decimal TYPE decimal(***30***,***15***);**
  e. Increase the precision of the decimal type in the source database.
  f. Insert data into the source database. The data can be properly written to Hudi.
  g. On the CDL web UI, start a task to update data in the source database.

Changing a Field Name

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. On the displayed page, click the hyperlink on the right of **CDLService UI** to go to the CDLService web UI. On the data synchronization job list page, locate the row that contains the target DDL change job, and choose **More** > **Stop** to stop a CDL job.

Step 2 Log in to the node where the client is installed as the client installation user and run the following commands:

cd *Client installation directory*

source bigdata_env

source Hudi/component_env

kinit *Component service user* (Skip this step if Kerberos authentication is disabled for the cluster.)

Step 3 Run the following commands to log in to the Spark SQL CLI:

cd Spark/spark/bin

./spark-sql

Step 4 Run the following statement on Hudi to change a field name:

ALTER TABLE *ddltest* **RENAME COLUMN** *columnName* **TO** *newColumnName;*

Step 5 Change the field name in the source database.

Step 6 On the CDL web UI, start a task to update data in the source database.

----End

Deleting a Field

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. On the displayed page, click the hyperlink on the right of **CDLService UI** to go to the CDLService web UI. On the data synchronization job list page, locate the row that contains the target DDL change job, and choose **More** > **Stop** to stop a CDL job.

Step 2 Log in to the node where the client is installed as the client installation user and run the following commands:

cd *Client installation directory*

source bigdata_env

source Hudi/component_env

kinit *Component service user* (Skip this step if Kerberos authentication is disabled for the cluster.)

Step 3 Run the following commands to log in to the Spark SQL CLI:

cd Spark/spark/bin

./spark-sql

Step 4 Run the following statement on Hudi to delete a field:

ALTER TABLE ddltest **DROP COLUMN** *columnName*;

Step 5 On the CDL web UI, start a task to update data in the source database.

----End

2.7 CDL Log Overview

Log Description

Log path: The default log storage path of CDL is **/var/log/Bigdata/cdl/***Role name abbreviation*.

- CDLService: **/var/log/Bigdata/cdl/service** (run logs) and **/var/log/Bigdata/audit/cdl/service** (audit logs).
- CDLConnector: **/var/log/Bigdata/cdl/connector** (run logs).

Log archive rule: The automatic CDL log compression function is enabled. By default, when the size of logs exceeds 50 MB, logs are automatically compressed into a log file named in the following format: *<Original log name>*.*<yyyy-mm-dd_hh-mm-ss>*.*[ID]*.**zip**. A maximum of 20 latest compressed files are reserved by default. The number of compressed files can be configured on Manager.

Table 2-79 Log list

Type	File	Description
Run logs	connect.log	CDLConnector run log.
	prestartDetail.log	Log that records cluster initialization before service startup.
	startDetail.log	Service startup log.
	stopDetail.log	Service stop log.
	cleanupDetail.log	Log that records the cleanup execution of services.
	check-serviceDetail.log	Log that records the verification of service status after service installation.
	cdl-db-operation.log	Log that records database initialization during service startup.
	cdl-app-launcher.log	Spark application startup log of CDL data synchronization tasks.
	cdl-dc-app-launcher.log	Spark application startup log of CDL data comparison tasks.
	serviceInstance-Check.log	Instance check log of CDLService.
	connectorInstance-Check.log	Instance check log of CDLConnector.

Type	File	Description
	ModifyDBPasswd.log	Log that records the resetting of the service database password.
	ranger-cdl-plugin-enable.log	Log that records the enabling or disabling of Ranger authentication.
	postinstallDetail.log	Service installation log.
	cdl_connector_pidxxx_gc.log.x	CDLConnector garbage collection (GC) log.
	cdl_service_pidxxx_gc.log.x	CDLService GC log.
	threadDump-CDLConnector-xxx.log	CDLConnector stack log.
	threadDump-CDLService-xxx.log	CDLService stack log.
Audit log	cdl-audit.log	Service audit log.

Log Level

Table 2-80 describes the log levels supported by CDL.

Levels of run logs are FATAL, ERROR, WARN, INFO, and DEBUG from the highest to the lowest priority. Run logs of equal or higher levels are recorded. The higher the specified log level, the fewer the logs recorded.

Table 2-80 Log levels

Type	Level	Description
Run log and audit log	FATAL	Logs of this level record fatal information about system.
	ERROR	Logs of this level record error information about system running.
	WARN	Logs of this level record exception information about the current event processing.
	INFO	Logs of this level record normal running status information about the system and events.

Type	Level	Description
	DEBUG	Logs of this level record system running and debugging information.

To modify log levels, perform the following operations:

Step 1 Go to the **All Configurations** page of CDL. For details, see **Modifying Cluster Service Configuration Parameters**.

Step 2 On the menu bar on the left, select the log menu of the target role.

Step 3 Select a desired log level.

Step 4 Save the configuration. In the displayed dialog box, click **OK** to make the configurations take effect.

NOTE

The configurations take effect immediately without the need to restart the service.

----End

Log Format

The following table lists the CDL log formats:

Table 2-81 Log formats

Type	Format	Example
Run log	*<yyyy-MM-dd HH:mm:ss,SSS>*|*<Log level>*|*<Name of the thread that generates the log>*|*<Message in the log>*|*<Location where the log event occurs>*	2021-06-15 17:25:19,658 | DEBUG | qtp2009591182-1754 | >fill SslConnection@5d04c5a0::SocketChannelEnd-Point@7c011c24{l=/10.244.224.65:21495,r=/10.244.224.83:53724,OPEN,fill=-,flush=-,to=1/30000}{io=0/0,kio=0,kro=1}->SslConnection@5d04c5a0{NOT_HANDSHAKING,eio=-1/-1,di=-1,fill=IDLE,flush=IDLE}~>DecryptedEndPoint@771f2f77{l=/10.244.224.65:21495,r=/10.244.224.83:53724,OPEN,fill=-,flush=-,to=19398/30000}=>HttpConnection@68c5859b[p=HttpParser{s=CONTENT,0 of -1},g=HttpGenerator@536e2de0{s=END}]=>HttpChannelOverHttp@7bf252bd{s=HttpChannelState@38be31e{s=IDLE rs=COMPLETED os=COMPLETED is=IDLE awp=false se=false i=false al=0},r=1,c=true/true,a=IDLE,uri=https://10.244.224.65:21495/api/v1/cdl/monitor/jobs/metrics,age=19382} | SslConnection.java:614

Type	Format	Example
Audit log	*<yyyy-MM-dd HH:mm:ss,SSS>*|*<Log level>*|*<Name of the thread that generates the log>*|*<Message in the log>*|*<Location where the log event occurs>*	2021-06-15 11:07:00,262 | INFO | qtp1846345504-30 | STARTTIME=2021-06-15 11:06:47.912 ENDTIME=2021-06-15 11:07:00.261 USERIP=10.144.116.198 USER=CDL User INSTANCE=10-244-224-65 OPERATION=Start CDL Job TARGET=CDCJobExecutionResource RESULT=SUCCESS | CDCAuditLogger.java:93

2.8 Common Issues About CDL

2.8.1 Hudi Does Not Receive Data After a CDL Job Is Executed

Symptom

After the CDL job for capturing data to Hudi is executed, related data exists in Kafka, but no record exists in Spark RDD, no related data exists in Hudi, and the error message "TopicAuthorizationException: No authorized to access topics" is displayed.

Possible Causes

The current user does not have the permission to consume Kafka data.

Procedure

Step 1 Log in to FusionInsight Manager and choose **System** > **Permission** > **User**. Locate the row containing the user who submitted the CDL job, click **Modify**, add the **kafkaadmin** user group, and click **OK**.

Step 2 On FusionInsight Manager, choose **Cluster** > **Services** > **CDL**. Click the hyperlink next to **CDLService UI** to access the CDLService web UI and restart the job.

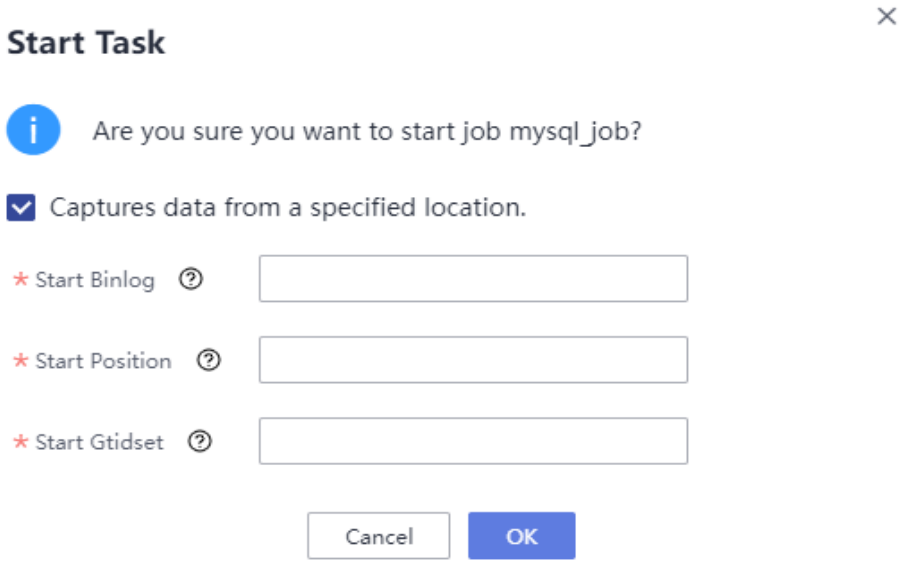
----End

2.8.2 How Do I Capture Data from a Specified Location When a MySQL Link Task Is Started?

Symptom

When a MySQL link task is started, data can be captured from a specified location. This section describes how to obtain parameter values of a specified location.

Figure 2-2 Start Task dialog box



Procedure

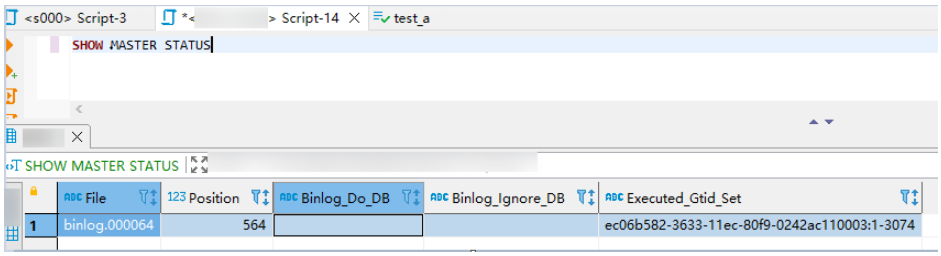
Step 1 Use a tool or CLI to connect to the MySQL database. In this example, Navicat is used.

Step 2 Run the following command:

SHOW MASTER STATUS

For example, on Navicat, choose **File** > **New Query** and enter **SHOW MASTER STATUS**. The command output is as follows:

Figure 2-3 SQL execution result



Step 3 In Figure 2-3, enter the value in the **File** column to **Start Binlog**, the value in the **Position** column to **Start Position**, and the value in the **Executed_Gtid_Set** column to **Start Gtidset**. Then, click **OK** to start the task.

NOTE

If there are two values (separated by a comma) of **Executed_Gtid_Set** in the query result in Step 2, set **Start Gtidset** to the first value. As shown in the following figure, the value of **Start Gtidset** is **13a90ad1-1f02-11ee-9ba9-fa163e6190d3:1-2794**.

File	Position	Binlog_Do_DB	Binlog_Ignore_DB	Executed_Gtid_Set
mysql-bin.000446	236			13a90ad1-1f02-11ee-9ba9-fa163e6190d3:1-2794,13d63a44-1f02-11ee-99c1-fa163e618eda:1-10766

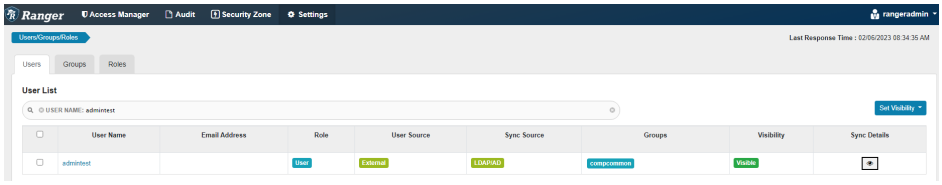
----End

2.8.3 Why Can a User Still Perform Operations on the Tasks Created by Itself After All Permissions of the User Are Deleted from Ranger?

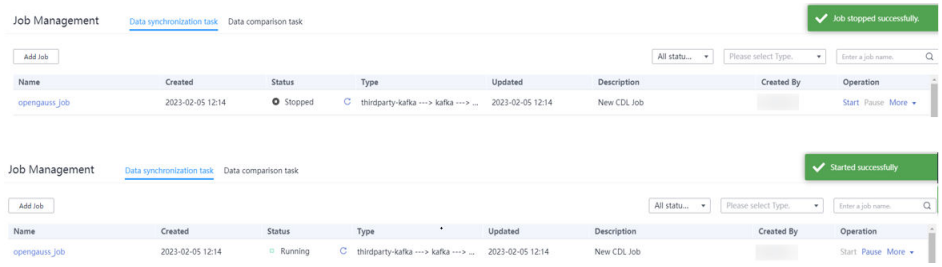
Symptom

In scenarios where Ranger authentication is enabled, after all permissions of a user are canceled, the user can still perform operations on the tasks created by itself. Example:

1. On the Ranger web UI, cancel all permissions of user **admintest**.



2. After logging in to the CDL web UI as user **admintest**, the user can still perform operations on the tasks created by itself on the **Job Management** page.



Possible Causes

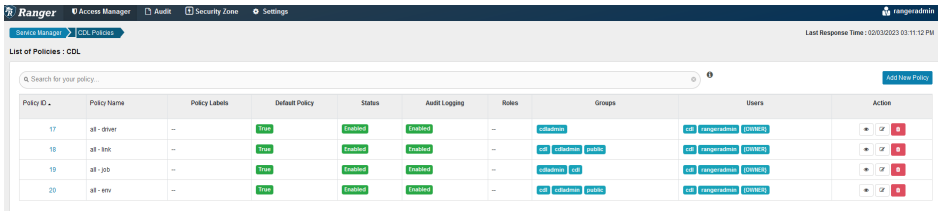
The **{OWNER}** permission is not deleted from the Ranger policy.

Procedure

Step 1 Log in to FusionInsight Manager as user **admin** and choose **Cluster** > **Services** > **Ranger**. On the page that is displayed, click the hyperlink next to **RangerAdmin UI** to access the Ranger web UI.

Step 2 On the Ranger web UI, click the username in the upper right corner, and choose **Log Out** to log out of the current user. Log in again as user **rangeradmin**.

Step 3 On the home page, click the component plug-in name in the **CDL** area, for example, **CDL**.



Step 4 Click in the **Action** column of each policy, delete user **{OWNER}** in the **Select User** column in the **Allow Conditions** area, and click **Save**.

Step 5 After 10 minutes, log in to the CDL web UI as the user whose **{OWNER}** permission has been deleted and attempt to perform operations on the jobs created by the user. It is found that the user does not have the permission to perform related operations.

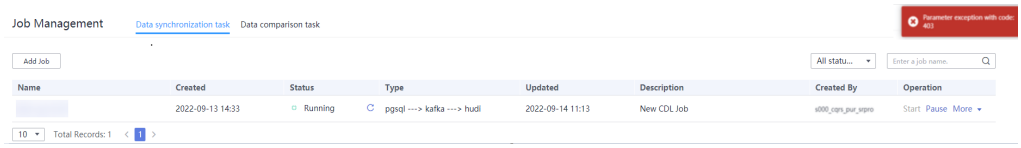
----End

2.9 CDL Troubleshooting

2.9.1 Error 403 Is Reported When a CDL Job Is Stopped

Symptom

The error message "parameter exception with code: 403" is displayed when a CDL job is stopped on the CDLService web UI.



Possible Causes

The current user does not have the permission to stop the job.

Procedure

Stop the job as the user who created the job. To view the job creator, log in to the CDLService web UI, locate the job in the job list, and view the creator in the **Creator** column.

2.9.2 Error 104 or 143 Is Reported After a CDL Job Runs for a Period of Time

Symptom

After an CDL job runs for a period of time, the YARN job fails and the status code **104** or **143** is returned. The following figure shows the returned status code 143.

Possible Causes

A large amount of data is captured to Hudi. As a result, the memory of the job is insufficient.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. Click the hyperlink next to **CDLService UI** to access the CDLService web UI. On the data synchronization job list page, locate the row that contains the target job and choose **More** > **Stop**. After the job is stopped, choose **More** > **Edit**.

Step 2 Change the value of **max.rate.per.partition** of Hudi to **6000** and click **Save**.

Step 3 On the data synchronization job list page, locate the row containing the target job and choose **More** > **Restart** to restart the job.

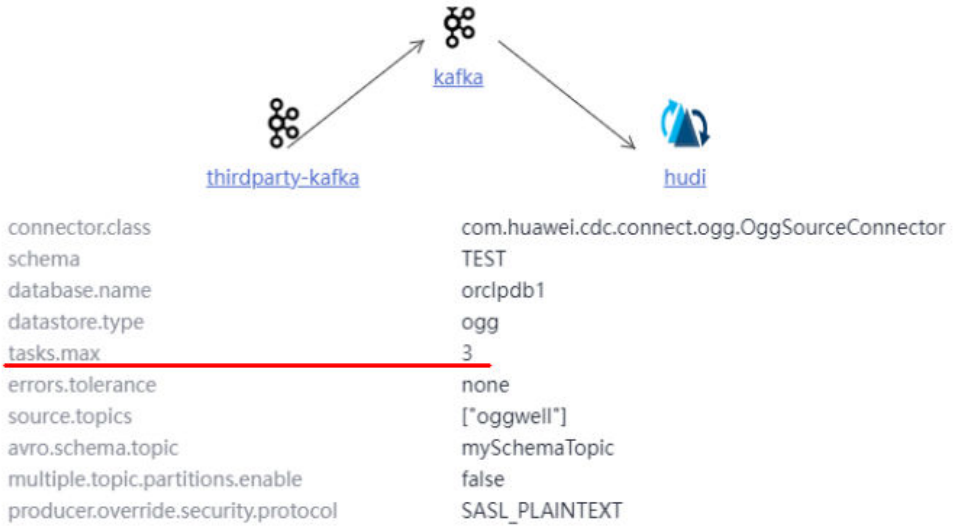
----End

2.9.3 Why Is the Value of Task configured for the OGG Source Different from the Actual Number of Running Tasks When Data Is Synchronized from OGG to Hudi?

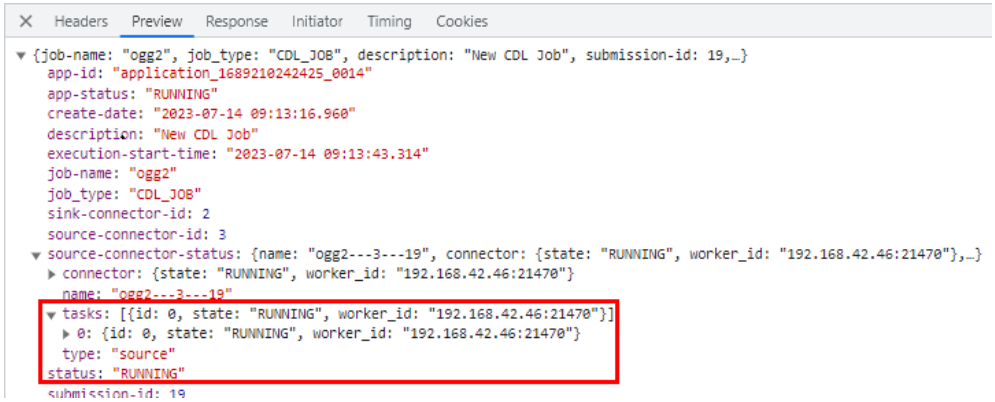
Symptom

When the CDL task for synchronizing data from OGG to Hudi is executed, the value of **tasks.max** specified in the source end (ThirdKafka) is different from the actual number of running tasks.

For example, on the CDL web UI, the value of **tasks.max** of the source job ThirdKafka is **3**.



However, the **tasks** information is **id: 0, state: xxx is**, indicating that there is only 1 task.



Possible Causes

The number of running tasks in OGG Source is calculated based on the smaller value between the values of **source.topics** and **tasks.max**.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. Click the hyperlink next to **CDLService UI** to access the CDLService web UI. On the data synchronization job list page, locate the row that contains the target job and choose **More** > **Stop**. After the job is stopped, choose **More** > **Edit**.

Step 2 Change the value of **tasks.max** of Thirdk Kafka to the number of topics specified by **source.topics** and click **Save**.

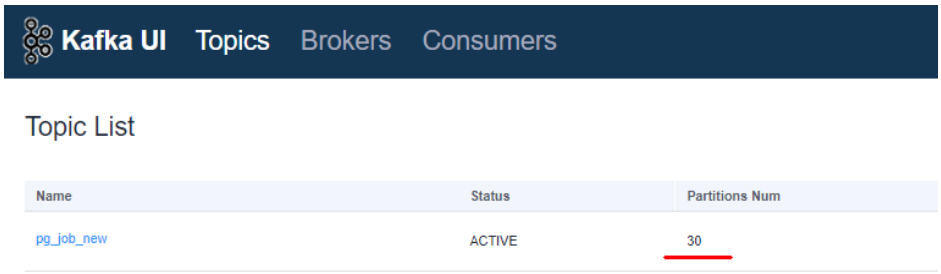
Step 3 On the data synchronization job list page, locate the row containing the target job and click **Start** to restart the job.

----End

2.9.4 Why Are There Too Many Topic Partitions Corresponding to the CDL Synchronization Task Names?

Symptom

After a CDL task is started, the value of **Partitions Num** of the CDL task name is too large in **Topic List** on the Kafka web UI.



Possible Causes

Topic Table Mapping is configured for CDL tasks, but the **WhiteList** parameter is not configured. CDL tasks of the schema configured for the CDL task have too many tables that are not synchronized. As a result, too many partitions are created for CDL task names.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. Click the hyperlink next to **CDLService UI** to access the CDLService web UI. On the data synchronization job list page, locate the row that contains the target job and choose **More** > **Stop**. After the job is stopped, choose **More** > **Edit**.

Step 2 Change the value of **WhiteList** of the Source to the configured number of Topic Table Mapping tables and click **Save**.

Step 3 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, and click the hyperlink on the right of **Kafka UI** to go to the Kafka web UI. On the **Topics** tab, search for the CDL task name and click **Action** > **Delete** in the **Operation** column.

Step 4 On the data synchronization job list page of CDL web UI, locate the row containing the target job and click **Start** to restart the job.

----End

2.9.5 What Should I When a CDL Task Is Executed to Synchronize Data to the Hudi, an Error Message Indicating that the Current User Does Not Have the Permission to Create Tables?

Symptom

After a CDL task is executed to synchronize data to Hudi, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Yarn**, and click **ResourceManager Web UI**

to access the Yarn web UI. In the task list, click the task ID, then **Logs**, and an error message is displayed, indicating that the current user does not have the permission to create a table. The error message is as follows:

```
org.apache.hadoop.hive.ql.security.authorization.plugin.HiveAccessControlException: Permission denied: Principal [name=xxx, type=USER] does not have following privileges for operation CREATETABLE [[CREATE] on Object [type=DATABASE, name=xxx]]
```

Possible Cause

The CDL service user does not have the permission to create tables in a database created by other users.

Procedure

Step 1 Log in to FusionInsight Manager, choose **System** > **Role** > **Add Role**, and enter the role name. In the **Configure Resource Permission** table, locate the cluster to be operated, choose **Hive** > **Hive Read and Write Permission**, select **Query**, **Delete**, **Insert**, **Create**, **Grant of Select**, **Grant of Delete**, **Grant of Insert**, and **Recursive**, and click **OK**.

Step 2 Click **User**, locate the user of the task, and click **Modify**. Add the role created in **Step 1**, and click **OK**.

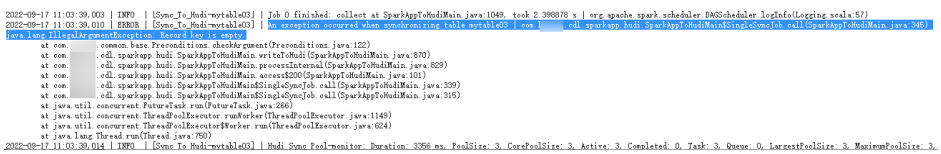
Step 3 Choose **Cluster** > **Services** > **CDL**. Click the hyperlink on the right of **CDLService UI** to go to the CDL web UI. In the row where the job is located, choose **More** > **Stop** to stop the CDL task. After the task is stopped, click **Start** to restart the task.

----End

2.9.6 Error Is Reported When the Job of Capturing Data From PgSQL to Hudi Is Started

Symptom

The error message "Record key is empty" is displayed when the job of capturing data from PgSQL to Hudi is started.



Possible Causes

The primary key parameter **table.primarykey.mapping** of the Hudi table is not configured.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **CDL**. Click the hyperlink next to **CDLService UI** to access the CDLService web UI. On the data synchronization job list page, locate the row that contains the target job and choose **More** > **Stop**. After the job is stopped, choose **More** > **Edit**.

Step 2 Configure the **table.primarykey.mapping** parameter on the Hudi table attribute configuration page and click **Save**. For details about the parameter, see **Table 2-36**.

Step 3 On the data synchronization job list page, locate the row containing the target job and click **Start** to restart the job.

----End

3 Using ClickHouse

3.1 ClickHouse Overview

ClickHouse Table Engines

Table engines play a key role in ClickHouse to determine:

- Where to write and read data
- Supported query modes
- Whether concurrent data access is supported
- Whether indexes can be used
- Whether multi-thread requests can be executed
- Parameters used for data replication

This section describes MergeTree and Distributed engine families, which are the most important and frequently used.

For details about other table engines, visit **https://clickhouse.tech/docs/en/engines/table-engines**.

- **MergeTree Family**

  Engines of the MergeTree family are the most universal and functional table engines for high-load tasks. They have the following key features:

  – Data is stored by partition and block based on partitioning keys.
  – Data index is sorted based on primary keys and the **ORDER BY** sorting keys.
  – Data replication is supported by table engines prefixed with "Replicated".
  – Data sampling is supported.

  When data is written, a table with this type of engine divides data into different folders based on the partitioning key. Each column of data in the folder is an independent file. A file that records serialized index sorting is created. This structure reduces the volume of data to be retrieved during data reading, greatly improving query efficiency.

– MergeTree

Syntax for creating a table:

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1] [TTL expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2] [TTL expr2],
    ...
    INDEX index_name1 expr1 TYPE type1(...) GRANULARITY value1,
    INDEX index_name2 expr2 TYPE type2(...) GRANULARITY value2
) ENGINE = MergeTree()
ORDER BY expr
[PARTITION BY expr]
[PRIMARY KEY expr]
[SAMPLE BY expr]
[TTL expr [DELETE|TO DISK 'xxx'|TO VOLUME 'xxx'], ...]
[SETTINGS name=value, ...]
```

Example:

```
CREATE TABLE default.test (
  name1 DateTime,
  name2 String,
  name3 String,
  name4 String,
  name5 Date,
  ...
) ENGINE = MergeTree()
PARTITION BY toYYYYMM(name5)
ORDER BY (name1, name2)
SETTINGS index_granularity = 8192
```

Parameters in the example are described as follows:

- **ENGINE = MergeTree()**: specifies the MergeTree engine.
- **PARTITION BY toYYYYMM(name4)**: specifies the partition. The sample data is partitioned by month, and a folder is created for each month.
- **ORDER BY**: specifies the sorting field. A multi-field index can be sorted. If the first field is the same, the second field is used for sorting, and so on.
- **index_granularity = 8192**: specifies the index granularity. One index value is recorded for every 8,192 data records.

If the data to be queried exists in a partition or sorting field, the data query time can be greatly reduced.

– ReplacingMergeTree

Different from MergeTree, ReplacingMergeTree deletes duplicate entries with the same sorting key. ReplacingMergeTree is suitable for clearing duplicate data to save space, but it does not guarantee the absence of duplicate data. Generally, it is not recommended.

Syntax for creating a table:

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = ReplacingMergeTree([ver])
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```


- SummingMergeTree

  When merging data parts in SummingMergeTree tables, ClickHouse merges all rows with the same primary key into one row that contains summed values for the columns with the numeric data type. If the primary key is composed in a way that a single key value corresponds to large number of rows, storage volume can be significantly reduced and the data query speed can be accelerated.

  Syntax for creating a table:

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = SummingMergeTree([columns])
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

  Example:

  Create a SummingMergeTree table named **testTable**.

```
CREATE TABLE testTable
(
    id UInt32,
    value UInt32
)
ENGINE = SummingMergeTree()
ORDER BY id
```

  Insert data into the table.

```
INSERT INTO testTable Values(5,9),(5,3),(4,6),(1,2),(2,5),(1,4),(3,8);
INSERT INTO testTable Values(88,5),(5,5),(3,7),(3,5),(1,6),(2,6),(4,7),(4,6),(43,5),(5,9),(3,6);
```

  Query all data in unmerged parts.

```
SELECT * FROM testTable
┌─id─┬─value─┐
│  1 │     6 │
│  2 │     5 │
│  3 │     8 │
│  4 │     6 │
│  5 │    12 │
└────┴───────┘
┌─id─┬─value─┐
│  1 │     6 │
│  2 │     6 │
│  3 │    18 │
│  4 │    13 │
│  5 │    14 │
│ 43 │     5 │
│ 88 │     5 │
└────┴───────┘
```

  If ClickHouse has not summed up all rows and you need to aggregate data by ID, use the **sum** function and **GROUP BY** statement.

```
SELECT id, sum(value) FROM testTable GROUP BY id
┌─id─┬─sum(value)─┐
│  4 │         19 │
│  3 │         26 │
│ 88 │          5 │
│  2 │         11 │
│  5 │         26 │
│  1 │         12 │
│ 43 │          5 │
└────┴────────────┘
```


Merge rows manually.

```
OPTIMIZE TABLE testTable
```

Query data in the **testTable** table again.

```
SELECT * FROM testTable
┌─id─┬─value─┐
│  1 │    12 │
│  2 │    11 │
│  3 │    26 │
│  4 │    19 │
│  5 │    26 │
│ 43 │     5 │
│ 88 │     5 │
└────┴───────┘
```

SummingMergeTree uses the **ORDER BY** sorting keys as the condition keys to aggregate data. That is, if sorting keys are the same, data records are merged into one and the specified merged fields are aggregated.

Data is pre-aggregated only when merging is executed in the background, and the merging execution time cannot be predicted. Therefore, it is possible that some data has been pre-aggregated and some data has not been aggregated. Therefore, the **GROUP BY** statement must be used during aggregation.

- AggregatingMergeTree

  AggregatingMergeTree is a pre-aggregation engine used to improve aggregation performance. When merging partitions, the AggregatingMergeTree engine aggregates data based on predefined conditions, calculates data based on predefined aggregate functions, and saves the data in binary format to tables.

  Syntax for creating a table:

  ```
  CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
  (
     name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
     name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
     ...
  ) ENGINE = AggregatingMergeTree()
  [PARTITION BY expr]
  [ORDER BY expr]
  [SAMPLE BY expr]
  [TTL expr]
  [SETTINGS name=value, ...]
  ```

  Example:

  You do not need to set the AggregatingMergeTree parameter separately. When partitions are merged, data in each partition is aggregated based on the **ORDER BY** sorting key. You can set the aggregate functions to be used and column fields to be calculated by defining the AggregateFunction type, as shown in the following example:

  ```
  create table test_table (
  name1 String,
  name2 String,
  name3 AggregateFunction(uniq,String),
  name4 AggregateFunction(sum,Int),
  name5 DateTime
  ) ENGINE = AggregatingMergeTree()
  PARTITION BY toYYYYMM(name5)
  ORDER BY (name1,name2)
  PRIMARY KEY name1;
  ```

  When data of the AggregateFunction type is written or queried, the ***state** and ***merge** functions need to be called. The asterisk (*) indicates

the aggregate functions used for defining the field type. For example, the **uniq** and **sum** functions are specified for the **name3** and **name4** fields defined in the **test_table**, respectively. Therefore, you need to call the **uniqState** and **sumState** functions and run the **INSERT** and **SELECT** statements when writing data into the table.

```
insert into test_table select '8','test1',uniqState('name1'),sumState(toInt32(100)),'2021-04-30 17:18:00';
insert into test_table select '8','test1',uniqState('name1'),sumState(toInt32(200)),'2021-04-30 17:18:00';
```

When querying data, you need to call the corresponding functions **uniqMerge** and **sumMerge**.

```
select name1,name2,uniqMerge(name3),sumMerge(name4) from test_table group by name1,name2;
┌─name1─┬─name2─┬─uniqMerge(name3)─┬─sumMerge(name4)─┐
│ 8     │ test1 │                1 │             300 │
└───────┴───────┴──────────────────┴─────────────────┘
```

AggregatingMergeTree is more commonly used together with materialized views, which are query views at the upper layer of other data tables. For details, visit **https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/aggregatingmergetree/**.

- CollapsingMergeTree

  CollapsingMergeTree defines a **Sign** field to record status of data rows. If **Sign** is **1**, the data in this row is valid. If **Sign** is **-1**, the data in this row needs to be deleted.

  Syntax for creating a table:

  ```
  CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
  (
     name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
     name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
     ...
  ) ENGINE = CollapsingMergeTree(sign)
  [PARTITION BY expr]
  [ORDER BY expr]
  [SAMPLE BY expr]
  [SETTINGS name=value, ...]
  ```

  Example:

  For details about the example, visit **https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/collapsingmergetree/**.

- VersionedCollapsingMergeTree

  The VersionedCollapsingMergeTree engine adds **Version** to the table creation statement to record the mapping between a **state** row and a **cancel** row in case that rows are out of order. The rows with the same primary key, same **Version**, and opposite **Sign** will be deleted during compaction.

  Syntax for creating a table:

  ```
  CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
  (
     name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
     name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
     ...
  ) ENGINE = VersionedCollapsingMergeTree(sign, version)
  [PARTITION BY expr]
  [ORDER BY expr]
  [SAMPLE BY expr]
  [SETTINGS name=value, ...]
  ```

  Example:

For details about the example, visit **https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/versionedcollapsingmergetree/**.

- GraphiteMergeTree

  The GraphiteMergeTree engine is used to store data in the time series database Graphite.

  Syntax for creating a table:

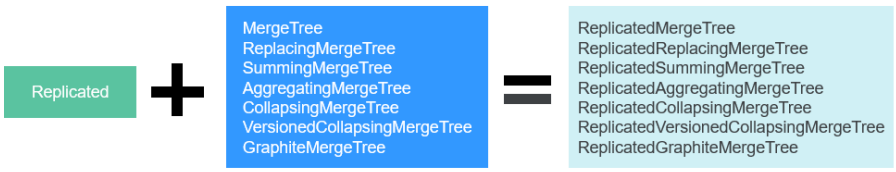
```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    Path String,
    Time DateTime,
    Value <Numeric_type>,
    Version <Numeric_type>
    ...
) ENGINE = GraphiteMergeTree(config_section)
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

  Example:

  For details about the example, visit **https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/graphitemergetree/**.

- **Replicated*MergeTree Engines**

  All engines of the MergeTree family in ClickHouse prefixed with Replicated become MergeTree engines that support replicas.



  Replicated series engines use ZooKeeper to synchronize data. When a replicated table is created, all replicas of the same shard are synchronized based on the information registered with ZooKeeper.

  Template for creating a Replicated engine:

```
ENGINE = Replicated*MergeTree('Storage path in ZooKeeper','Replica name', ...)
```

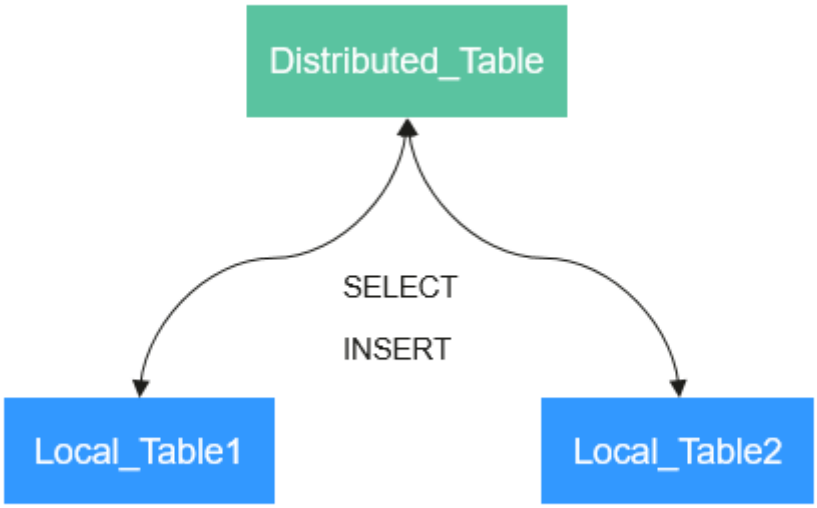
  Two parameters need to be specified for a Replicated engine:

  - *Storage path in ZooKeeper*: specifies the path for storing table data in ZooKeeper. The path format is ***/clickhouse/tables/{shard}/Database name/Table name***.
  - Replica name: Generally, **{replica}** is used.

- **Distributed Table Engines**

  The Distributed engine does not store any data. It serves as a transparent proxy for data shards and can automatically transmit data to each node in the cluster. Distributed tables need to work with other local data tables. Distributed tables distribute received read and write tasks to each local table where data is stored.

Figure 3-1 Distributed



Template for creating a Distributed engine:

```
ENGINE = Distributed(cluster_name, database_name, table_name, [sharding_key])
```

Parameters of a distributed table are described as follows:

- **cluster_name**: specifies the cluster name. When a distributed table is read or written, the cluster configuration information is used to search for the corresponding ClickHouse instance node.
- **database_name**: specifies the database name.
- **table_name**: specifies the name of a local table in the database. It is used to map a distributed table to a local table.
- **sharding_key** (optional): specifies the sharding key, based on which a distributed table distributes data to each local table.

Example:

```
-- Create a ReplicatedMergeTree local table named test.
CREATE TABLE default.test ON CLUSTER default_cluster_1
(
    `EventDate` DateTime,
    `id` UInt64
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/test', '{replica}')
PARTITION BY toYYYYMM(EventDate)
ORDER BY id

-- Create a distributed table named test_all based on the local table test.
CREATE TABLE default.test_all ON CLUSTER default_cluster_1
(
    `EventDate` DateTime,
    `id` UInt64
)
ENGINE = Distributed(default_cluster_1, default, test, rand())
```

Rules for creating a distributed table:

- When creating a distributed table, add **ON CLUSTER** *cluster_name* to the table creation statement so that the statement can be executed once on

a ClickHouse instance and then distributed to all instances in the cluster for execution.

- Generally, a distributed table is named in the following format: *Local table name*_all. It forms a one-to-many mapping with local tables. Then, multiple local tables can be operated using the distributed table proxy.
- Ensure that the structure of a distributed table is the same as that of local tables. If they are inconsistent, no error is reported during table creation, but an exception may be reported during data query or insertion.

ClickHouse Data Types

Table 3-1 describes the data types used in the ClickHouse component of MRS.

For details about the ClickHouse data types, see **ClickHouse Data Types**.

Table 3-1 Data types

Category	Keyword	Type	Description
Data type	Int8	Int8	Value range: -128 to 127
	Int16	Int16	Value Range: -32768 to 32767
	Int32	Int32	Value range: -2147483648 to 2147483647
	Int64	Int64	Value range: -9223372036854775808 to 9223372036854775807
Floating point type	Float32	Float	Similar to the Float type in C, a single-precision floating point number occupies 4 bytes in storage of a computer and is described in 32-bit binary mode.
	Float64	Double	Similar to the Double type in the C language, a double-precision floating point number occupies eight bytes in the machine and is described in 64-bit binary mode.

Category	Keyword	Type	Description
Decimal type	Decimal	Decimal	A signed fixed-point number that can maintain precision during addition, subtraction, and multiplication operations. The following formats are supported: • Decimal(P, S) • Decimal32(S) • Decimal64(S) • Decimal128(S) **NOTE** • P: precision. Valid range: [1:38]. It determines the number of decimal digits (including fractions) that can be contained. • S: scale. The value ranges from 0 to P, which determines the number of decimal places in the decimal part of a number.
String type	String	String	The character string can be of any length. It can contain any set of bytes, including empty bytes. Therefore, the String type can replace the VARCHAR, BLOB, and CLOB types in other database management systems.
	FixedString	Fixed-length string	When the length of the data happens to be N bytes, the FixedString type is efficient. In other cases, this may reduce efficiency. Examples of values that can be effectively stored in columns of the FixedString type: • IP address represented in binary • Language code (ru_RU, en_US...) • Currency code (RUB...) • Hash value represented in binary mode (FixedString (16) for MD5 and FixedString (32) for SHA256)

Category	Keyword	Type	Description
Time and date type	Date	Date	A Date value takes up two bytes, indicating the date value from 1970-01-01 (unsigned) to the current time. Date values are stored without the time zone.
	DateTime	Timestamp	A Unix timestamp value takes up four bytes (unsigned). Value range of this type is the same as the Date type. The minimum value is 1970-01-01 00:00:00. Timestamp values are accurate to seconds. Leap seconds are not supported. The system time zone will be used when the client or server is started.
	DateTime64	DateTime64	This type allows you to store both the date and time of a specific point in time.
Boolean type	Boolean	Boolean	ClickHouse does not support the Boolean type. You can use the UInt8 type for Boolean values. Valid values are 0 and 1.
Array type	Array	Array	An Array value is a collection of elements of the same data type. The elements can be of a random data type, even the Array type itself. However, multi-dimensional arrays are not recommended, because ClickHouse supports multi-dimensional arrays only to a limited extent. For example, you cannot store multi-dimensional arrays in MergeTree tables.

Category	Keyword	Type	Description
Tuple type	Tuple	Tuple	A Tuple value is a collection of elements of different data types. Tuple values cannot be stored in tables, except for memory tables. You can use Tuple values to group temporary columns. In queries, you can use **IN** expressions and **lambda** functions with specific parameters to group temporary columns.
Domains data type	Domains	Domains	The implementation of the Domains type varies based on different values: If the values are IPv4 addresses, the Domains type is binary compatible with the UInt32 type. Compared with the UInt32 type, the Domains type saves the binary storage space and supports more readable input and output formats.
Enumerated type	Enum8	Enum8	Value range: -128 to 127 An Enum value stores the mapping of 'string'= integer, for example, **Enum8('hello' = 1, 'world' = 2)**.
	Enum16	Enum16	Value Range: -32768 to 32767
Nullable type	Nullable	Nullable	Unless otherwise stated in ClickHouse server configurations, the default value of the NULLABLE type is NULL. Nullable values cannot be included in table indexes. Nullable values can be stored together with the normal values of TypeName. For example, columns of the Nullable(Int8) type can store values of the Int8 type, while rows without values store NULL.

Category	Keyword	Type	Description
Nested type	nested	nested	A nested data structure is similar to a table inside a cell. You can specify the parameters of a nested data structure, such as field name and data type, the same way that you specify parameters in a **CREATE TABLE** statement. Each row in a **CREATE TABLE** statement can correspond to a random number of rows in a nested data structure. Example: **Nested (Name1 Type1,Name2 Type2, ...)**

3.2 ClickHouse User Permission Management

3.2.1 ClickHouse User Rights

User Permission Model

ClickHouse user permission management enables unified management of users, roles, and permissions on each ClickHouse instance in the cluster. You can use the permission management module of the Manager UI to create users, create roles, and bind the ClickHouse access permissions. User permissions are controlled by binding roles to users.

- Resource management: **Table 3-2** lists the resources supported by ClickHouse permission management.
- Resource permissions: **Table 3-3** lists the resource permissions supported by ClickHouse.

Table 3-2 Permission management objects supported by ClickHouse

Resource	Integration	Remarks
Database	Yes (level 1)	-
Table	Yes (level 2)	-
View	Yes (level 2)	Same as tables

Table 3-3 Resource permission list

Resource	Available Permission	Remarks
Database	CREATE	CREATE DATABASE/TABLE/ VIEW/DICTIONARY
Table/View	SELECT/INSERT	-

3.2.2 Creating a ClickHouse Role

This topic describes how to create and configure a ClickHouse role on Manager as an MRS cluster administrator. You can grant the administrator permission and permissions to read and write tables and databases to the ClickHouse role.

Prerequisites

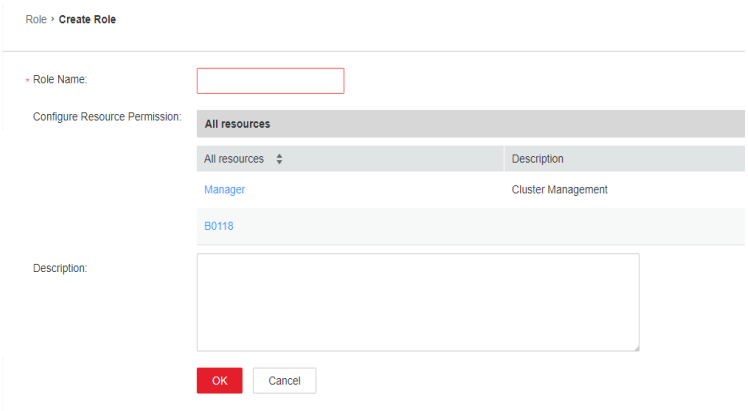
- The ClickHouse and Zookeeper services are running properly.
- The **ON CLUSTER** statement has been used to create a database or table in the cluster to ensure that the metadata of the database and table on each ClickHouse node is the same.

NOTE

After the permission is granted, it takes about 1 minute for the permission to take effect.

Adding the ClickHouse Role

Step 1 Log in to Manager and choose **System** > **Permission** > **Role**. On the **Role** page, click **Create Role**.



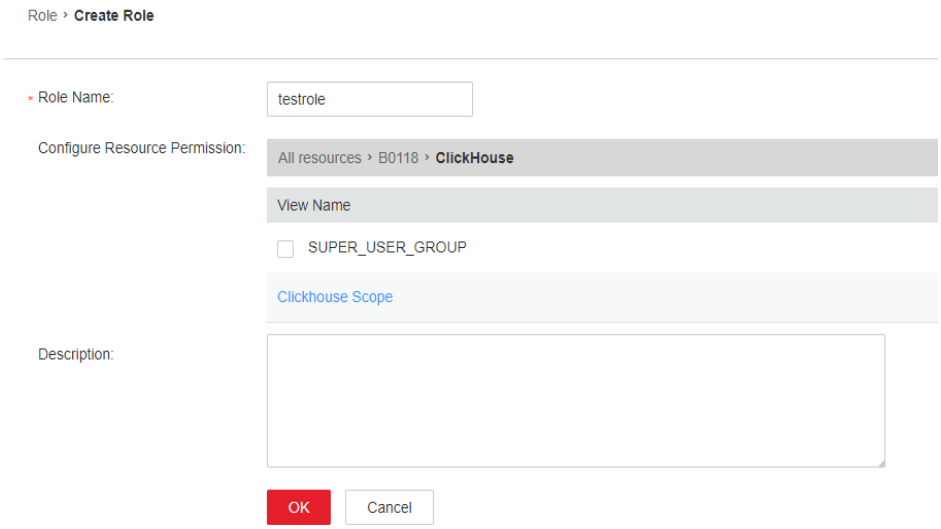
Step 2 On the **Create Role** page, specify **Role Name**. In the **Configure Resource Permission** area, click the cluster name. On the service list page that is displayed, click the ClickHouse service.

Determine whether to create a role with the ClickHouse administrator permissions based on service requirements.

NOTE

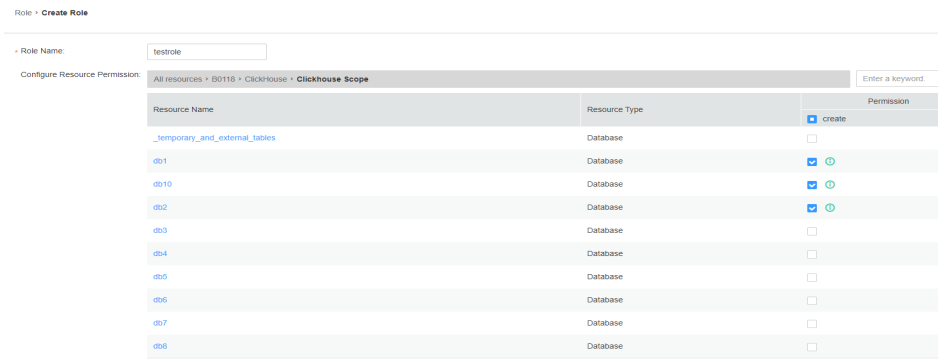
- The ClickHouse administrator has all the database operation permissions except the permissions to create, delete, and modify users and roles.
- Only the built-in user **clickhouse** of ClickHouse has the permission to manage users and roles.

- If yes, go to **Step 3**.
- If no, go to **Step 4**.



Step 3 Select **SUPER_USER_GROUP** and click **OK**.

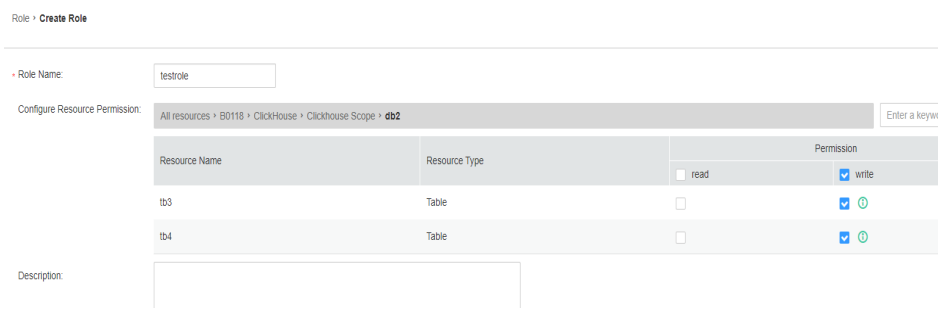
Step 4 Click **Clickhouse Scope**. The ClickHouse database resource list is displayed. If you select **create**, the role has the create permission on the database.



Determine whether to grant the permission based on the service requirements.

- If yes, click **OK**.
- If no, go to **Step 5**.

Step 5 Click the resource name and select the *Database resource name to be operated*. On the displayed page, select **READ** (SELECT permission) or **WRITE** (INSERT permission) based on service requirements, and click **OK**.



----End

Adding a User and Binding the ClickHouse Role to the User

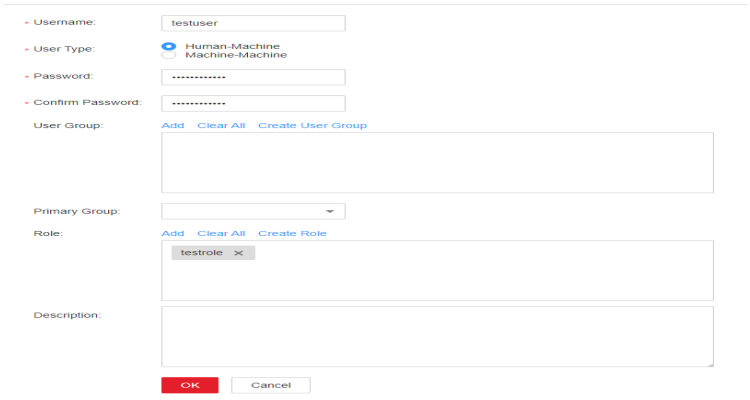
Step 1 Log in to Manager and choose **System** > **Permission** > **User** and click **Create**.

Step 2 Select **Human-Machine** for **User Type** and set **Password** and **Confirm Password** to the password of the user.

NOTE

- Username: The username cannot contain hyphens (-). Otherwise, the authentication will fail.
- Password: The password cannot contain special characters $, ., and #. Otherwise, the authentication will fail.

Step 3 In the **Role** area, click **Add**. In the displayed dialog box, select a role with the ClickHouse permission and click **OK** to add the role. Then, click **OK**.



Step 4 Log in to the node where the ClickHouse client is installed and use the new username and password to connect to the ClickHouse service.

- Run the following command to go to the client installation directory:

  cd /opt/*Client installation directory*

- Run the following command to configure environment variables:

  source bigdata_env

- If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the current user. The user must have the permission to create ClickHouse tables. Therefore, you need to bind the corresponding role to the user. For details, see **Adding the ClickHouse Role**. If Kerberos authentication is disabled for the current cluster, skip this step.

kinit *User added in* ***Step 1***

- Log in to the system as the new user.

  Cluster with Kerberos authentication disabled:

  clickhouse client --host *IP address of the ClickHouse instance* **--multiline --port** *ClickHouse port number* **--secure**

  Cluster with Kerberos authentication enabled:

  clickhouse client --host *IP address of the ClickHouse instance*--**user** *Username* **--password --port** 9440 **--secure**

  Enter the user password.

  NOTE

  The user in normal mode is the **default** user, or you can create an administrator using the open source capability provided by the ClickHouse community. You cannot use the users created on FusionInsight Manager.

----End

Granting Permissions Using the Client in Abnormal Scenarios

By default, the table metadata on each node of the ClickHouse cluster is the same. Therefore, the table information on a random ClickHouse node is collected on the permission management page of Manager. If the **ON CLUSTER** statement is not used when databases or tables are created on some nodes, the resource may fail to be displayed during permission management, and permissions may not be granted to the resource. To grant permissions on the local table on a single ClickHouse node, perform the following steps on the background client.

NOTE

The following operations are performed based on the obtained roles, database or table names, and IP addresses of the node where the corresponding ClickHouseServer instance is located.

- To obtain the IP address of the ClickHouseServer instance, log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse**, and click the **Instances** tab.
- The default system domain name is **hadoop.com**. Log in to FusionInsight Manager and choose **System** > **Permission** > **Domain and Mutual Trust**. The value of **Local Domain** is the system domain name. Change the letters to lowercase letters when running a command.

Step 1 Log in to the node where the ClickHouseServer instance is located as user **root**.

Step 2 Run the following command to obtain the path of the **clickhouse.keytab** file:

ls ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/keytab/clickhouse.keytab

Step 3 Log in to the node where the client is installed as the client installation user.

Step 4 Run the following command to go to the client installation directory:

cd /opt/client

Step 5 Run the following command to configure environment variables:

source bigdata_env

Step 6 Run the following command to connect to the ClickHouseServer instance:

If Kerberos authentication is enabled for the current cluster, run the following command:

clickhouse client --host *IP address of the node where the ClickHouseServer instance is located* **--user clickhouse/hadoop.***<System domain name>* **--password** *clickhouse.keytab path obtained in* ***Step 2*** **--port** *ClickHouse port number* **--secure**

If Kerberos authentication is disabled for the current cluster, run the following command:

clickhouse client --host *IP address of the node where the ClickHouseServer instance is located* **--user clickhouse --port** *ClickHouse port number*

Step 7 Run the following statement to grant permissions to a database:

In the syntax for granting permissions, *DATABASE* indicates the name of the target database, and *role* indicates the target role.

GRANT [ON CLUSTER *cluster_name***]** *privilege* **ON** *{DATABASE|TABLE}* **TO** *{user | role}*

For example, grant user **testuser** the CREATE permission on database **t2**:

GRANT CREATE ON *t2* **to** *testuser***;**

Step 8 Run the following commands to grant permissions on the table or view. In the following command, *TABLE* indicates the name of the table or view to be operated, and *user* indicates the role to be operated.

Run the following command to grant the query permission on tables in a database:

GRANT SELECT ON *TABLE* **TO** *user;*

Run the following command to grant the write permission on tables in a database:

GRANT INSERT ON *TABLE* **TO** *user***;**

NOTE

For details about ClickHouse **GRANT** operations and permission description, visit **https://clickhouse.tech/docs/en/sql-reference/statements/grant/**.

Step 9 Run the following command to exit the client:

quit;

----End

3.2.3 Configuring Interconnection Between ClickHouse and OpenLDAP Authentication System

ClickHouse can be interconnected with OpenLDAP. You can manage accounts and permissions in a centralized manner by adding the OpenLDAP server configuration and creating users on ClickHouse. You can use this method to import users from the OpenLDAP server to ClickHouse in batches.

This section applies only to MRS 3.1.0 or later.

Prerequisites

- The MRS cluster and ClickHouse instances are running properly, and the ClickHouse client has been installed.
- OpenLDAP has been installed and is running properly.

Creating a ClickHouse User for Interconnecting with the OpenLDAP Server

Step 1 Log in to Manager and choose **Cluster** > **Services** > **ClickHouse**. Click the **Configurations** tab and then **All Configurations**.

Step 2 Choose **ClickHouseServer(Role)** > **Customization**, and add the following OpenLDAP configuration parameters to the **clickhouse-config-customize** configuration item. See **Figure 3-2**.

Table 3-4 OpenLDAP parameters

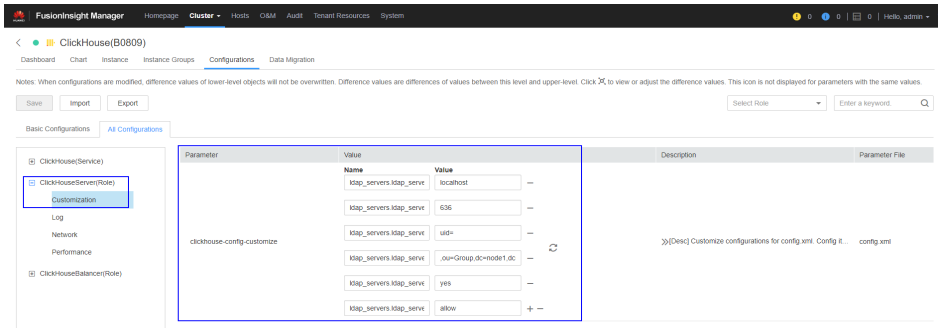
Parameter	Description	Example Value
ldap_servers.ldap_server_name.host	OpenLDAP server host name or IP address. This parameter cannot be empty.	localhost
ldap_servers.ldap_server_name.port	OpenLDAP server port number. If **enable_tls** is set to **true**, the default port number is **636**. Otherwise, the default port number is **389**.	636
ldap_servers.ldap_server_name.auth_dn_prefix	Prefix and suffix used to construct the DN to bind to. The generated DN will be constructed as a string in the following format: **auth_dn_prefix** + **escape(user_name)** + **auth_dn_suffix**. Use a comma (,) as the first non-space character of **auth_dn_suffix**.	uid=
ldap_servers.ldap_server_name.auth_dn_suffix		,ou=Group,dc=node1,dc=com
ldap_servers.ldap_server_name.enable_tls	A tag to trigger the use of the secure connection to the OpenLDAP server. • Set it to **no** for the plaintext (ldap://) protocol (not recommended). • Set it to **yes** for the LDAP over SSL/TLS (ldaps://) protocol.	yes

Parameter	Description	Example Value
ldap_servers.ldap_server_name.tls_require_cert	SSL/TLS peer certificate verification behavior. The value can be **never**, **allow**, **try**, or **require**.	allow

NOTE

For details about other parameters, see **<ldap_servers> Parameters**.

Figure 3-2 OpenLDAP configuration



Step 3 After the configuration is complete, click **Save**. In the displayed dialog box, click **OK**. After the configuration is saved, click **Finish**.

Step 4 On Manager, click **Instance**, select a ClickHouseServer instance, and choose **More** > **Restart Instance**. In the displayed dialog box, enter the password and click **OK**. In the displayed **Restart instance** dialog box, click **OK**. Confirm that the instance is restarted successfully as prompted and click **Finish**.

Step 5 Log in to the ClickHouseServer instance node and go to the **${BIGDATA_HOME}/FusionInsight_ClickHouse_***Version number*_*x_x*_**ClickHouseServer/etc** directory.

cd ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/*x_x*_ClickHouseServer/etc

Step 6 Run the following command to view the **config.xml** configuration file and check whether the OpenLDAP parameters are configured successfully:

cat config.xml

```
[root@k                3 etc]# cat config.xml
<yandex>
  <ldap_servers>
    <ldap_server_name>
      <auth_dn_prefix>uid=</auth_dn_prefix>
      <port>636</port>
      <host>localhost</host>
      <enable_tls>yes</enable_tls>
      <tls_require_cert>allow</tls_require_cert>
      <auth_dn_suffix>,ou=Group,dc=node1,dc=com</auth_dn_suffix>
    </ldap_server_name>
  </ldap_servers>
  <zookeeper>
```

Step 7 Log in to the node where the ClickHouseServer instance is located as user **root**.

Step 8 Run the following command to obtain the path of the **clickhouse.keytab** file:

ls ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/keytab/clickhouse.keytab

Step 9 Log in to the node where the client is installed as the client installation user.

Step 10 Run the following command to go to the ClickHouse client installation directory:

cd /opt/client

Step 11 Run the following command to configure environment variables:

source bigdata_env

Step 12 Run the following command to connect to the ClickHouseServer instance:

- If Kerberos authentication is enabled for the current cluster, use **clickhouse.keytab** to connect to the ClickHouseServer instance.

  clickhouse client --host *IP address of the node where the ClickHouseServer instance is located* **--user clickhouse/hadoop.***<System domain name>* **--password** *clickhouse.keytab path obtained in* ***Step 8*** **--port** *ClickHouse port number*

  NOTE

  The default system domain name is **hadoop.com**. Log in to FusionInsight Manager and choose **System** > **Permission** > **Domain and Mutual Trust**. The value of **Local Domain** is the system domain name. Change the letters to lowercase letters when running a command.

- If Kerberos authentication is not enabled for the current cluster, connect to the ClickHouseServer instance as the ClickHouse administrator.

  clickhouse client --host *IP address of the node where the ClickHouseServer instance is located* **--user clickhouse --port** *ClickHouse port number*

Step 13 Create a common user of OpenLDAP.

Run the following statement to create user **testUser** in cluster **default_cluster** and set **ldap_server** to the OpenLDAP server name in the **<ldap_servers>** tag in **Step 6**. In this example, the name is **ldap_server_name**.

CREATE USER *testUser* **ON CLUSTER** *default_cluster* **IDENTIFIED WITH ldap_server BY** 'ldap_server_name'**;**

testUser indicates an existing username in OpenLDAP. Change it based on the site requirements.

Step 14 Log out of the client, and then log in to the client as the new user to check whether the configuration is successful.

exit;

clickhouse client --host *IP address of the ClickHouseServer instance* **--user** *testUser* **--password --port** *ClickHouse port number*

Enter the password of testUser.

----End

<ldap_servers> Parameters

- **host**: OpenLDAP server host name or IP address. This parameter is mandatory.
- **port**: port number of the OpenLDAP server. If **enable_tls** is set to **true**, the default value is **636**. Otherwise, the value is **389**.
- **auth_dn_prefix, auth_dn_suffix**: Prefix and suffix used to construct the DN to bind to. The generated DN will be constructed as a string in the following format: **auth_dn_prefix** + **escape(user_name)** + **auth_dn_suffix**. Use a comma (,) as the first non-space character of **auth_dn_suffix**.
- **enable_tls**: tag to trigger the use of the secure connection to the OpenLDAP server.

  Set it to **no** for the plaintext (ldap://) protocol (not recommended).

  Set it to **yes** for LDAP over SSL/TLS (ldaps://) protocol (recommended and default).
- **tls_minimum_protocol_version**: minimum protocol version of SSL/TLS

  The value can be **ssl2**, **ssl3**, **tls1.0**, **tls1.1**, or **tls1.2** (default).
- **tls_require_cert**: SSL/TLS peer certificate verification behavior

  The value can be **never**, **allow**, **try**, or **require** (default).
- **tls_cert_file**: certificate file
- **tls_key_file**: certificate key file
- **tls_ca_cert_file**: CA certificate file
- **tls_ca_cert_dir**: directory where the CA certificate is stored
- **tls_cipher_suite**: cipher suites allowed

3.3 ClickHouse Client Practices

ClickHouse is a column-based database oriented to online analysis and processing. It supports SQL query and provides good query performance. The aggregation analysis and query performance based on large and wide tables is excellent, which is one order of magnitude faster than other analytical databases.

ClickHouse uses the ReplicatedMergeTree engine and ZooKeeper to replicate tables. When creating a table, you can specify the engine to enable table replication for high availability. Shards and replicas of each table are independent from each other.

ClickHouse supports distributed tables by using the Distributed engine. Views are created on all shards (local tables) to ease distributed queries. ClickHouse supports data sharding, which is one of the features of distributed storage. This allows for parallel read and write for faster queries.

This topic uses the cluster client to quickly connect to the ClickHouse service in a cluster.

Prerequisites

- The cluster client has been installed, for example, in the **/opt/client** directory.
- For clusters with Kerberos authentication enabled, you have created a user with ClickHouse operation permissions on Manager, for example, **clickhouseuser**.

Using a ClickHouse Client

Step 1 Install the cluster client. For details, see **Installing a Client**.

Step 2 Log in to the node where the client is installed as the client installation user.

Step 3 Run the following command to go to the client installation directory:

cd /opt/client

Step 4 Run the following command to configure environment variables:

source bigdata_env

Step 5 If Kerberos authentication has been enabled for the cluster, authenticate the user first. If not, skip this step.

The service user must have the permission to create ClickHouse tables. For details, see **Creating a ClickHouse Role**. Create a user on Manager and associate the user to the required role.

kinit *Component service user*

The following provides an example:

kinit *clickhouseuser*

Step 6 Run the **clickhouse client** command to connect to the ClickHouse server.

- Using a non-SSL method for login when Kerberos authentication is disabled for the ClickHouse cluster

  clickhouse client --host *IP address of the ClickHouse instance* **--port** *9000* **--user** *Username* **--password**

- Using SSL for login when Kerberos authentication is enabled for the current cluster:

  clickhouse client --host *IP address of the ClickHouse instance* **--port** *9440* **--user** *Username* **--password**

Table 3-5 Parameters of the **clickhouse client** command

Parameter	Description
--host	Name of the server node. You can use the host name or IP address of the node where the ClickHouse instance is deployed. To obtain the IP address of the ClickHouseServer instance, log in to FusionInsight Manager of the cluster and choose **Cluster** > **Services** > **ClickHouse** > **Instances**.

Parameter	Description
--port	Connection port. • Clusters with Kerberos authentication enabled use SSL connections by default. The default port is 9440, and the **--secure** parameter must be specified. You can also obtain the port number from the **tcp_port_secure** parameter of the ClickHouseServer instance. If a non-SSL connection is required in this scenario, log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse** > **Configurations**, search for **SSL_NONESSL_BOTH_ENABLE**, change the value to **true**, and restart all ClickHouse instances. • Clusters with Kerberos authentication disabled use non-SSL connections by default. The default port is 9000, and the **--secure** parameter is not required. You can also obtain the port number from the **tcp_port** parameter of the ClickHouseServer instance. If an SSL connection is required in this scenario, log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse** > **Configurations**, search for **SSL_NONESSL_BOTH_ENABLE**, change the value to **true**, and restart all ClickHouse instances. **NOTE** The preceding ports are open-source ports. If a custom port policy is used during cluster creation, replace the ports by referring to **Common Troubleshooting**. You can run the **clickhouse -h** command to view the command help of the ClickHouse component.
--user	Connection username. • If Kerberos authentication has been enabled for the cluster (the cluster is in security mode) and you have passed the kinit authentication, the **--user** and **--password** parameters are optional for client login. You must create a user with this name on Manager because there is no default user for Kerberos authentication. • If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), do not use the ClickHouse user created on FusionInsight Manager to log in to the client. You need to execute the **create user SQL** statement on the client to create a ClickHouse user. Alternatively, you can use the **default** user for login.
--password	User password for connection. This parameter is used together with the **--user** parameter.
--query	Query to process when using non-interactive mode.
--database	The current database is used by default. This parameter uses the default configuration on the server.

Parameter	Description
--multiline	If this parameter is specified, multiline queries are allowed. (**Enter** only indicates line feed and does not indicate that the query statement is complete.)
--multiquery	If this parameter is specified, multiple queries separated with semicolons (;) can be processed. This parameter is valid only in non-interactive mode.
--format	Specified default format used to output the result.
--vertical	If this parameter is specified, the result is output in vertical format by default. In this format, each value is printed on a separate line, which helps to display a wide table.
--time	If this parameter is specified, the query execution time is printed to **stderr** in non-interactive mode.
--stacktrace	If this parameter is specified, stack trace information will be printed when an exception occurs.
--config-file	Name of the configuration file.
--secure	If this parameter is specified, the server will be connected in SSL mode.
--history_file	Path of files that record command history.
--param_<name>	Query with parameters. Pass values from the client to the server. For details, see **https://clickhouse.tech/docs/en/interfaces/cli/#cli-queries-with-parameters**.

Step 7 Run the **quit;** command to exit the ClickHouse server connection.

----End

Viewing Environment Parameters (cluster) of ClickHouse

Step 1 Use the ClickHouse client to connect to the ClickHouse server.

Step 2 Query the **cluster** identifier and other information about the environment parameters.

select cluster,shard_num,replica_num,host_name from system.clusters;

```
SELECT
    cluster,
    shard_num,
    replica_num,
    host_name
FROM system.clusters

┌─cluster─────────────┬─shard_num─┬─replica_num─┬─host_name──────────┐
│ default_cluster_1   │         1 │           1 │ node-master1dOnG   │
│ default_cluster_1   │         1 │           2 │ node-group-1tXED0001 │
│ default_cluster_1   │         2 │           1 │ node-master2OXQS   │
│ default_cluster_1   │         2 │           2 │ node-group-1tXED0002 │
│ default_cluster_1   │         3 │           1 │ node-master3QsRI   │
```



```
┌─default_cluster_1─┬─3─┬─2─┬─node-group-1tXED0003─┐
└───────────────────┴───┴───┴──────────────────────┘

6 rows in set. Elapsed: 0.001 sec.
```

Step 3 Query the **shard** and **replica** identifiers.

select * from system.macros;

```
SELECT *
FROM system.macros

┌─macro───────┬─substitution─┐
│ id          │ 76           │
│ replica     │ 2            │
│ shard       │ 3            │
└─────────────┴──────────────┘

3 rows in set. Elapsed: 0.001 sec.
```

----End

Creating a Local Replicated Table and a Distributed Table

Step 1 Create a replicated table using the ReplicatedMergeTree engine.

For details about the syntax, see **https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/replication/#creating-replicated-tables**.

For example, to create the ReplicatedMergeTree table **test** in the **default** database on all **default_cluster_1** nodes, run the following command:

CREATE TABLE *default.test* **ON CLUSTER** *default_cluster_1*

(

`EventDate` DateTime,

`id` UInt64

)

ENGINE = ReplicatedMergeTree('*/clickhouse/tables/{shard}/default/test***', '***{replica}***')**

PARTITION BY toYYYYMM(EventDate)

ORDER BY id;

- **ON CLUSTER** is a distributed DDL clause that creates the same local table on all instances in the cluster in one execution.
- **default_cluster_1** is the cluster identifier used for viewing environment parameters of the ClickHouse service.

CAUTION

ReplicatedMergeTree engine receives the following two parameters:

- Storage path of the table data in ZooKeeper

  The path must be in the **/clickhouse** directory. Otherwise, data insertion may fail due to insufficient ZooKeeper quota.

  To avoid data conflict between different tables in ZooKeeper, the directory must be in the following format:

  /clickhouse/tables/{shard}/default/test, in which **/clickhouse/tables/{shard}** is fixed, *default* indicates the database name, and *text* indicates the name of the created table.

- Replica name: Generally, **{replica}** is used.

```
CREATE TABLE default.test ON CLUSTER default_cluster_1
(
   `EventDate` DateTime,
   `id` UInt64
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/test', '{replica}')
PARTITION BY toYYYYMM(EventDate)
ORDER BY id

┌─host──────────────────────┬─port─┬─status─┬─error─┬─num_hosts_remaining─┬─num_hosts_activ
e─┐
│ node-group-1tXED0002      │ 9000 │      0 │       │                   5 │               3 │
│ node-group-1tXED0003      │ 9000 │      0 │       │                   4 │               3 │
│ node-master1dOnG          │ 9000 │      0 │       │                   3 │               3 │
└───────────────────────────┴──────┴────────┴───────┴─────────────────────┴─────────────────┘
┌─host──────────────────────┬─port─┬─status─┬─error─┬─num_hosts_remaining─┬─num_hosts_activ
e─┐
│ node-master3QsRI          │ 9000 │      0 │       │                   2 │               0 │
│ node-group-1tXED0001      │ 9000 │      0 │       │                   1 │               0 │
│ node-master2OXQS          │ 9000 │      0 │       │                   0 │               0 │
└───────────────────────────┴──────┴────────┴───────┴─────────────────────┴─────────────────┘

6 rows in set. Elapsed: 0.189 sec.
```

Step 2 Create a distributed table using the Distributed engine.

For example, run the following statement to create the **test_all** table in the **default** database on all **default_cluster_1** nodes:

CREATE TABLE *default.test_all* **ON CLUSTER** *default_cluster_1*

(

`EventDate` DateTime,

`id` UInt64

)

ENGINE = Distributed(*default_cluster_1, default, test, rand()***);**

```
CREATE TABLE default.test_all ON CLUSTER default_cluster_1
(
   `EventDate` DateTime,
   `id` UInt64
)
ENGINE = Distributed(default_cluster_1, default, test, rand())

┌─host──────────────────────┬─port─┬─status─┬─error─┬─num_hosts_remaining─┬─num_hosts_activ
```

```
e┐
│ node-group-1tXED0002 │ 9000 │ 0 │ │ 5 │ 0 │
│ node-master3QsRI │ 9000 │ 0 │ │ 4 │ 0 │
│ node-group-1tXED0003 │ 9000 │ 0 │ │ 3 │ 0 │
│ node-group-1tXED0001 │ 9000 │ 0 │ │ 2 │ 0 │
│ node-master1dOnG │ 9000 │ 0 │ │ 1 │ 0 │
│ node-master2OXQS │ 9000 │ 0 │ │ 0 │ 0 │
└──────────────────┴──────┴───┴──┴──┴──┘
```

6 rows in set. Elapsed: 0.115 sec.

NOTE

Distributed requires the following parameters:

- **default_cluster_1** is the cluster identifier used for viewing environment parameters of the ClickHouse service.
- **default** indicates the name of the database where the local table is located.
- **test** is the name of the local table.
- (Optional) Sharding key

  This key and the weight configured in the **config.xml** file determine the route for writing data to the distributed table, that is, the physical table to which the data is written. It can be the original data (for example, **site_id**) of a column in the table or the result of the function call, for example, **rand()** is used in the preceding SQL statement. Note that data must be evenly distributed in this key. Another common operation is to use the hash value of a column with a large difference, for example, **intHash64(user_id)**.

----End

ClickHouse Table Data Operations

Step 1 Insert data to the local table.

For example, insert data to the **test** table.

insert into test values(toDateTime(now()), rand());

Step 2 Query local table information.

For example, query data in the **test** table.

select * from test;

```
SELECT *
FROM test

┌─────────EventDate─┬──────────id─┐
│ 2020-11-05 21:10:42 │ 1596238076 │
└─────────────────────┴────────────┘

1 rows in set. Elapsed: 0.002 sec.
```

Step 3 Query the distributed table.

The following example queries the distributed table**test_all**, which is created based on **test**, and gets the same result as querying the **test** table.

select * from test_all;

```
SELECT *
FROM test_all
```



```
┌─────────────EventDate─┬──────────id─┐
│ 2020-11-05 21:10:42   │ 1596238076  │
└───────────────────────┴─────────────┘

1 rows in set. Elapsed: 0.004 sec.
```

Step 4 Switch to the shard node with the same **shard_num** and query the information about the current table. The same table data can be queried.

For example, run the **exit;** command to exit the original node.

Run the following command to switch to the **node-group-1tXED0003** node:

clickhouse client --host *node-group-1tXED0003* **--multiline --port 9440 --secure;**

show tables;

```
SHOW TABLES

┌─name─────┐
│ test     │
│ test_all │
└──────────┘
```

Step 5 Query local table data. For example, query data in the **test** table on the **node-group-1tXED0003** node.

select * from test;

```
SELECT *
FROM test

┌─────────────EventDate─┬──────────id─┐
│ 2020-11-05 21:10:42   │ 1596238076  │
└───────────────────────┴─────────────┘

1 rows in set. Elapsed: 0.005 sec.
```

Step 6 Switch to the shard node with different **shard_num** value and query the data of the created table.

For example, exit the **node-group-1tXED0003** node.

exit;

Switch to the **node-group-1tXED0001** node.

clickhouse client --host *node-group-1tXED0001* **--multiline --port 9440 --secure;**

Query the local table **test**. The **test** table data cannot be queried on the different shard nodes.

select * from test;

```
SELECT *
FROM test

Ok.
```

Query data in the distributed table **test_all**. The data can be queried.

select * from test_all;

```
SELECT *
FROM test
```



```
┌─────────────EventDate─┬──────────id─┐
│ 2020-11-05 21:12:19   │ 3686805070  │
└───────────────────────┴─────────────┘

1 rows in set. Elapsed: 0.002 sec.
```

----End

Common Troubleshooting

After the command for connecting to the ClickHouse client is executed, error message "Connection refused" is displayed.

Check whether the cluster uses custom ports (**Component Port** is set to **Custom** during cluster creation). If custom ports are used, replace the ports used in the command for connecting to the ClickHouse component client with the default custom ports in the following table.

Parameter	Default Open Source Port	Default Custom Port	Port Description
interserver_http_port	9009	9009	HTTP port for the communication between ClickHouse servers.
interserver_https_port	9010	9010	HTTPS port for the communication between ClickHouse servers.
http_port	8123	8123	Port for connecting to the ClickHouse server through HTTP.
https_port	8443	8443	Port for connecting to the ClickHouse server through HTTPS.
tcp_port	9000	9000	Port for connecting the client to the ClickHouse server through TCP.
tcp_port_secure	9440	9440	Port for connecting the client to the ClickHouse server through TCP SSL.
lb_tcp_port	21424	21424	TCP port listened by ClickHouseBalancer
lb_http_port	21425	21425	HTTP port listened by ClickHouseBalancer
lb_https_port	21426	21426	HTTPS port listened by ClickHouseBalancer
lb_tcp_secure_port	21428	21428	TCP SSL port listened by ClickHouseBalancer

3.4 ClickHouse Data Import

3.4.1 Interconnecting ClickHouse with RDS for MySQL

ClickHouse provides efficient data analysis in OLAP scenarios. It can map a table on the remote database server to the ClickHouse cluster through a database engine such as MySQL, so data can be analyzed in the ClickHouse cluster. The following describes how to interconnect the ClickHouse cluster with the MySQL database instance of RDS.

Prerequisites

- You have prepared the RDS database instance to be interconnected with and the username and password of the database. For details, see **Creating and Connecting to an RDS DB Instance**.
- A ClickHouse cluster has been created and is running properly.

Constraints

- The RDS database instance and ClickHouse cluster are in the same VPC and subnet.
- Before synchronizing data, you need to evaluate the impact on the performance of the source and destination databases. You are advised to synchronize data during off-peak hours.
- Currently, ClickHouse can interconnect with MySQL and PostgreSQL instances of RDS, but cannot interconnect with SQL Server instances.

Interconnecting ClickHouse with RDS Using the MySQL Engine

The MySQL engine is used to map tables on the remote MySQL server to ClickHouse and allows you to run INSERT and SELECT statements on tables to facilitate data exchange between ClickHouse and MySQL.

Syntax for using the MySQL engine:

```
CREATE DATABASE [IF NOT EXISTS] db_name [ON CLUSTER cluster]
ENGINE = MySQL('host:port', ['database' | database], 'user', 'password')
```

Parameters of the MySQL engine:

- **host:port**: IP address and port number of the RDS MySQL database instance.
- **database**: Name of the RDS MySQL database.
- **user**: Username of the RDS MySQL database.
- **password**: Password of the RDS MySQL database user.

Example of using the MySQL engine:

Step 1 Connect to the MySQL database of RDS. For details, see **Connecting to a DB Instance**.

Step 2 Create a table in the MySQL database and insert data into the table.

Create table **mysql_table**.

CREATE TABLE `mysql_table` (

`int_id` INT NOT NULL AUTO_INCREMENT,

`float` FLOAT NOT NULL,

PRIMARY KEY (`int_id`));

Insert data into the table.

insert into mysql_table (`int_id`, `float`) VALUES (1,2);

Step 3 Log in to the node where the ClickHouse client is installed. Run the following command to go to the client installation directory:

cd /opt/client

Step 4 Run the following command to configure environment variables:

source bigdata_env

Step 5 If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the current user. The user must have the permission to create ClickHouse tables. Therefore, you need to bind the corresponding role to the user. For details, see **Creating a ClickHouse Role**. If Kerberos authentication is disabled for the current cluster, skip this step.

1. Run the following command if it is an MRS 3.1.0 cluster:

   export CLICKHOUSE_SECURITY_ENABLED=true

2. **kinit** *Component service user*

   Example: **kinit clickhouseuser**

Step 6 Run the client command to connect to ClickHouse.

clickhouse client --host *IP address of the ClickHouse instance* **--user** *Username* **--password --port** *Port number*

Enter the user password.

Step 7 Create a MySQL database in ClickHouse. After the database is created, it automatically exchanges data with a MySQL server.

CREATE DATABASE mysql_db ENGINE = MySQL('*IP address of the RDS MySQL database instance***:***Port number of the MySQL database instance***', '***MySQL database name***', '***MySQL database username***', '***Password of the MySQL database user***');**

Step 8 Switch to the created database **mysql_db** and query data in the table.

USE mysql_db;

Query the table data in the MySQL database in ClickHouse.

SELECT * FROM mysql_table;

```
┌─int_id─┬─float─┐
│      1 │     2 │
└────────┴───────┘
```

Data can be properly queried after being inserted.

INSERT INTO mysql_table VALUES (3,4);

SELECT * FROM mysql_table;

```
┌─int_id─┬─float─┐
│      1 │     2 │
│      3 │     4 │
└────────┴───────┘
```

----End

Enabling mysql_port for ClickHouse

Configure the ClickHouse port to connect the MySQL client to ClickHouse.

NOTE

This operation is available for MRS 3.1.2 only.

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** then **All Configurations**. Search for the **clickhouse-config-customize** parameter and add the following configuration: **Name**: **mysql_port** and **Value**: **9004**.

NOTE

The value is customizable.

Click **Save**.

Step 2 Click the **Dashboard** tab, click **More**, and select **Restart Instance**; alternatively, click **More** and select **Instance Rolling Restart**.

----End

3.4.2 Interconnecting ClickHouse with OBS

Using S3 Table Functions

Step 1 Log in to the active OMS node.

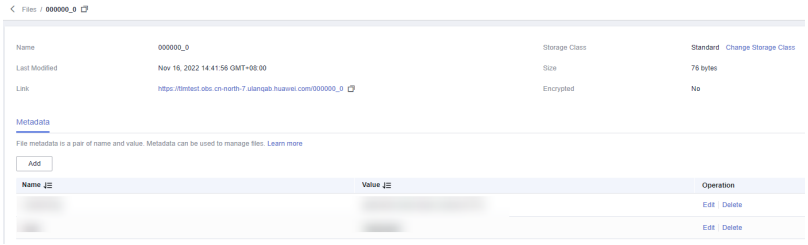
Step 2 Run the following command to obtain the data stored in OBS:

select * from S3(*path, [ak, sk,] format, structure, [compression]***)**

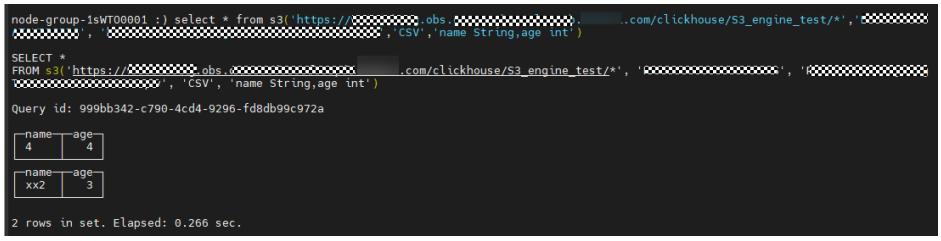
NOTE

- *path*: *Access domain name*/*OBS file path*. Log in to OBS console and choose **Parallel File Systems**. On the page that is displayed, click the file system name. On the **Files** tab page, click the file name. The value of **Link** is the path.

  Figure 3-3 File path



- **ak**: Optional. The AK that has the permission to access OBS.
- **sk**: Optional. The SK that has the permission to access OBS.
- **format**: The file format.
- **structure**: The table schema.
- **compression**: Optional. The compression type.



----End

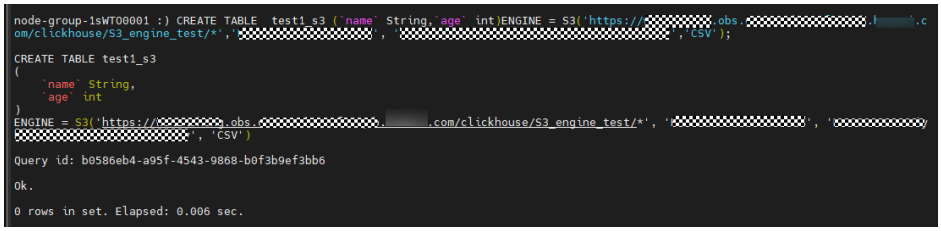
Using the S3 Table Engine

Step 1 Log in to the active OMS node.

Step 2 Run the following commands to create a table:

CREATE TABLE *test1_s3* **(***'name' String, 'age' int***)**

ENGINE = S3(*path, [ak, sk,] format, [compression]***)**



Step 3 Run the following command to query the table:

select * from *test1_s3*;

----End

Modifying Manager Configurations

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** then **All Configurations**. Search for the **clickhouse-config-customize** parameter and add values. The following table lists the parameter values.

Parameter	Value
s3.endpoint-name.endpoint	OBS bucket address
s3.endpoint-name.access_key_id	OBS AK. For details on how to obtain the AK, see **How Do I Obtain the Access Key AK/SK?**
s3.endpoint-name.secret_access_key	OBS SK. For details about how to obtain the OBS SK, see **How Do I Obtain the AK/SK?**

Step 2 Typically, the URL shared by OBS contains HTTPS. If the URL is inaccessible directly, perform the following operations to modify the configuration:

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** then **All Configurations**. Search for the **clickhouse-config-customize** parameter and add values. The following table lists the parameter values.

Parameter	Value
openSSL.client.loadDefaultCAFile	true
openSSL.client.cacheSessions	true
openSSL.client.disableProtocols	sslv2,sslv3
openSSL.client.preferServerCiphers	true
openSSL.client.invalidCertificateHandler.name	AcceptCertificateHandler

Step 3 After the modification, click **Save**.

----End

3.4.3 Interconnecting ClickHouse with HDFS (MRS 3.2.0-LTS)

NOTE

This topic is available for MRS 3.2.0-LTS version only.

Scenario

Connect ClickHouse to HDFS to read and write files.

Prerequisites

- The ClickHouse client has been installed in a directory, for example, **/opt/client**.
- A user, for example, **clickhouseuser**, who has permissions on ClickHouse tables and has the permission to access HDFS has been created on FusionInsight Manager.
- A corresponding directory exists in HDFS. The HDFS engine of ClickHouse only works with files but does not create or delete directories.
- Only the ClickHouse cluster deployed on x86 nodes can connect to HDFS. The ClickHouse cluster deployed on Arm nodes cannot connect to HDFS.

Procedure

Step 1 Log in to the node where the client is installed as the client installation user.

Step 2 Run the following command to go to the client installation directory:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Run the following command to authenticate the current user. (Change the password upon the first authentication. Skip this step for a cluster with Kerberos authentication disabled.)

kinit clickhouseuser

Step 5 Run the client command of ClickHouse to log in to the ClickHouse client.

clickhouse client --host *Service IP address of the ClickHouseServer instance* **--secure --port 9440**

Step 6 Run the following command to connect ClickHouse to HDFS:

CREATE TABLE default.hdfs_engine_table (`name` String, `value` UInt32) ENGINE = HDFS('hdfs://*{namenode_ip}***:***{dfs.namenode.rpc.port}***/tmp/secure_ck.txt', 'TSV')**

NOTE

- To obtain the service IP address of the ClickHouseServer instance, perform the following steps:

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. On the page that is displayed, click the **Instances** tab. In this tab, obtain the service IP addresses of the ClickHouseServer instance.

- To obtain the value of *namenode_ip*, perform the following steps:

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS**. On the page that is displayed, click the **Instances** tab. In this tab, obtain the service IP addresses of the active NameNode.

- To obtain the value of *dfs.namenode.rpc.port*, perform the following steps:

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS**. On the page that is displayed, click the **Configurations** tab then the **All Configurations** sub-tab. In this sub-tab, search for **dfs.namenode.rpc.port** to obtain its value.

- HDFS file path to be accessed:

  If multiple files need to be accessed, add an asterisk (*) to the end of the folder, for example, **hdfs://***{namenode_ip}*:***{dfs.namenode.rpc.port}***/tmp/***.

----End

3.4.4 Interconnecting ClickHouse with HDFS (MRS 3.3.0-LTS or later)

NOTE

This section applies to MRS 3.3.0-LTS or later.

Scenario

This section describes how to read and write files after connecting ClickHouse in security mode to HDFS in security mode. Functions same as those provided in the open source community are available after ClickHouse is connected to HDFS in normal mode. ClickHouse and HDFS deployed in clusters of different modes cannot be connected.

Prerequisites

- The ClickHouse client has been installed in a directory, for example, **/opt/client**.
- A user, for example, **clickhouseuser**, who has permissions on ClickHouse tables and has the permission to access HDFS has been created on FusionInsight Manager.
- A corresponding directory exists in HDFS. The HDFS engine of ClickHouse only works with files but does not create or delete directories.
- When ClickHouse accesses HDFS across clusters, a user, for example, **hdfsuser**, who has the permission to access HDFS has been created on FusionInsight Manager in the cluster where HDFS is.
- You have obtained the HDFS cluster domain name by logging in to FusionInsight Manager and choosing **System** > **Permission** > **Domain and Mutual Trust**.

- ClickHouse cannot connect to encrypted HDFS directories.

Interconnecting ClickHouse with HDFS in a Cluster

Step 1 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS**, select **Configuration** > **All Configurations**, search for and change the value of **hadoop.rpc.protection** to **Authentication** or **Integrity**, save the settings, and restart the HDFS service.

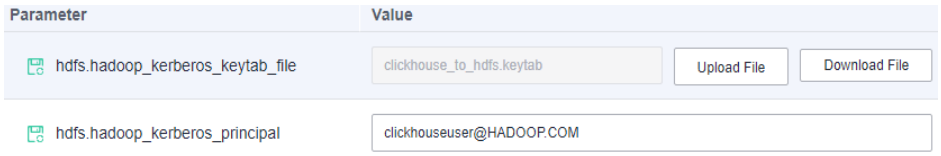
Step 2 Choose **System** > **User**, select **clickhouseuser**, and choose **More** > **Download Authentication Credential**.

NOTE

For the first authentication, change the initial password before downloading the authentication credential file. Otherwise, the security authentication will fail.

Step 3 Decompress the downloaded authentication credential package and change the name of **user.keytab** to **clickhouse_to_hdfs.keytab**.

Step 4 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse**, and click **Configurations** then **All Configurations**. Click **ClickHouseServer(Role)** and select **Engine**. Click **Upload File** next to **hdfs.hadoop_kerberos_keytab_file**. Then upload the authentication credential file in **Step 3**. Set **hdfs.hadoop_kerberos_principal** to a value in the format of *Username*@*Domain name*, for example, **clickhouseuser@HADOOP.COM**.



If you are connecting ClickHouse to the HDFS in HA mode, perform the following steps:

1. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** > **Instances**, click any instance name, and click the **hdfs-site.xml** in the **Configuration File** area of the **Dashboard** tab.
2. Obtain the values of the following parameters from **hdfs-site.xml**. For example, if two NameServices are configured for HDFS, one is **hacluster** and the other is **ns1**, you need to obtain the values of the following parameters:

   dfs.nameservices, **dfs.ha.namenodes.***, **dfs.namenode.rpc-address.*.*** and **dfs.client.failover.proxy.provider.*** (change the value to **org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider**)

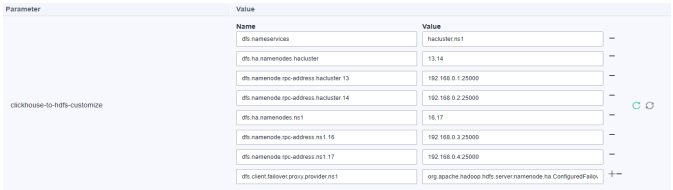
NOTICE

- Value of **dfs.namenode.rpc-address.*.*** can be obtained from the HDFS. If the value is a host name, change it to the service IP address of the host name.
- When configuring **clickhouse-to-hdfs-customize** parameters, use lowercase letters for NameService (regardless of the original letter case of the NameService). For example, if a NameService of the HDFS cluster is **NS1**, convert it to **ns1** in the **clickhouse-to-hdfs-customize** file.

The following is an example:

```
<property>
<name>dfs.nameservices</name>
<value>hacluster,ns1</value>
</property>
<property>
<name>dfs.namenode.rpc-address.hacluster.13</name>
<value>192.168.0.1:25000</value>
</property>
<property>
<name>dfs.namenode.rpc-address.hacluster.14</name>
<value>192.168.0.2:25000</value>
</property>
<property>
<name>dfs.ha.namenodes.hacluster</name>
<value>13,14</value>
</property>
<name>dfs.namenode.rpc-address.ns1.16</name>
<value>192.168.0.3:25000</value>
</property>
<property>
<name>dfs.namenode.rpc-address.ns1.17</name>
<value>192.168.0.4:25000</value>
</property>
<property>
<name>dfs.ha.namenodes.ns1</name>
<value>16,17</value>
</property>
<property>
<name>dfs.client.failover.proxy.provider.ns1</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
<property>
<name>dfs.client.failover.proxy.provider.hacluster</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
```

3. On FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse** > **Configurations** > **All Configurations** > **ClickHouseServer(Role)** > **Engine**, and add the parameter values obtained in **Step 4.2** to the custom configuration **clickhouse-to-hdfs-customize**.



Step 5 Save the configuration and restart ClickHouse.

Step 6 Log in to the node where the client is installed as the client installation user.

Step 7 Run the following command to go to the client installation directory:

cd /opt/client

Step 8 Run the following command to configure environment variables:

source bigdata_env

Step 9 Run the following command to authenticate the current user. (Skip this step for a cluster with Kerberos authentication disabled.)

kinit clickhouseuser

Step 10 Run the client command of ClickHouse to log in to the ClickHouse client.

clickhouse client --host *Service IP address of the ClickHouseServer instance* **--secure --port 9440**

NOTE

- To obtain the service IP address of the ClickHouseServer instance, perform the following steps:

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. On the page that is displayed, click the **Instance** tab. On this tab page, obtain the service IP addresses of the ClickHouseServer instance.

Step 11 Connect ClickHouse to HDFS.

- Run the following command to connect to the HDFS component in HA mode:

  CREATE TABLE default.hdfs_engine_table (`name` String, `value` UInt32) ENGINE = HDFS('hdfs://*{nameservice}***/tmp/secure_ck.txt', 'TSV')**

- Run the following command to connect to the HDFS component in non-HA mode:

  CREATE TABLE default.hdfs_engine_table (`name` String, `value` UInt32) ENGINE = HDFS('hdfs://*{namenode_ip}***:***{dfs.namenode.rpc.port}***/tmp/secure_ck.txt', 'TSV')**

NOTE

- To obtain the value of *nameservice*, perform the following steps:

  On FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** > **NameService Management** and obtain the value of **NameService**. The **nameservice** used in the table creation statement must be in lowercase (regardless of whether the value in the HDFS cluster is in lowercase). For example, if the value in the HDFS cluster is **NS1**, the value in the table creation statement must be in lowercase **ns1**.

- To obtain the value of *namenode_ip*, perform the following steps:

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS**. On the page that is displayed, click the **Instance** tab. On this tab page, obtain the service IP addresses of the active NameNode.

- To obtain the value of *dfs.namenode.rpc.port*, perform the following steps:

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS**. On the page that is displayed, click the **Configurations** tab then the **All Configurations** sub-tab. On this sub-tab page, search for **dfs.namenode.rpc.port** to obtain its value.

- HDFS file path to be accessed:

  If multiple files need to be accessed, add an asterisk (*) to the end of the folder, for example, **hdfs://***{namenode_ip}*:*{dfs.namenode.rpc.port}*/**tmp/***.

  ClickHouse cannot connect to encrypted HDFS directories.

- Write data. For details, see **Process of Writing ClickHouse Data to HDFS**.

----End

Interconnecting ClickHouse with HDFS Across Clusters

Step 1 Log in to the FusionInsight Manager of the HDFS cluster, choose **Cluster** > **Services** > **HDFS**, select **Configuration** > **All Configurations**, search for and change the value of **hadoop.rpc.protection** to **Authentication** or **Integrity**, save the settings, and restart the HDFS service.

Step 2 Log in to FusionInsight Manager of the ClickHouse cluster and choose **System** > **Permission** > **Domain and Mutual Trust**. Configure mutual trust or unilateral mutual trust with the HDFS cluster. To configure unilateral mutual trust, configure mutual trust with the HDFS cluster only on the ClickHouse cluster.

Step 3 Log in to FusionInsight Manager of the HDFS cluster and choose **System** > **Permission** > **User**. On the page that is displayed, select **hdfsuser**, click **More**, and select **Download Authentication Credential**.

NOTE

For the first authentication, change the initial password before downloading the authentication credential file. Otherwise, the security authentication will fail.

Step 4 Decompress the downloaded authentication credential package and change the name of **user.keytab** to **clickhouse_to_hdfs.keytab**.

Step 5 Log in to FusionInsight Manager of the ClickHouse cluster, choose **Cluster** > **Services** > **ClickHouse**, and click **Configurations** then **All Configurations**. Click **ClickHouseServer(Role)** and select **Engine**. Click **Upload File** next to **hdfs.hadoop_kerberos_keytab_file** to upload the authentication credential file in **Step 3**. Set **hdfs.hadoop_kerberos_principal** to a value in the format of *Username*@*Domain name*, for example, **hdfsuser@HDFS_HADOOP.COM**.

If you are connecting ClickHouse to the HDFS in HA mode, perform the following steps:

1. Log in to FusionInsight Manager of the HDFS cluster, choose **Cluster** > **Service** > **HDFS** > **Instances**, click any instance name, and click the **hdfs-site.xml** in the **Configuration File** area in the **Dashboard** tab.
2. Obtain the values of the following parameters from **hdfs-site.xml**. For example, if two NameServices are configured for HDFS, one is **hacluster** and the other is **ns1**, you need to obtain the values of the following parameters:

   dfs.nameservices, **dfs.ha.namenodes.***, **dfs.namenode.rpc-address.*.*** and **dfs.client.failover.proxy.provider.*** (change the value to **org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider**)

NOTICE

- Value of **dfs.namenode.rpc-address.*.*** can be obtained from the HDFS. If the value is a host name, change it to the service IP address of the host name.
- When configuring **clickhouse-to-hdfs-customize** parameters, use lowercase letters for NameService (regardless of the original letter case of the NameService). For example, if a NameService of the HDFS cluster is **NS1**, convert it to **ns1** in the **clickhouse-to-hdfs-customize** file.

The following is an example:

```
<property>
<name>dfs.nameservices</name>
<value>hacluster,ns1</value>
</property>
<property>
<name>dfs.namenode.rpc-address.hacluster.13</name>
<value>192.168.0.1:25000</value>
</property>
<property>
<name>dfs.namenode.rpc-address.hacluster.14</name>
<value>192.168.0.2:25000</value>
</property>
<property>
<name>dfs.ha.namenodes.hacluster</name>
<value>13,14</value>
</property>
<name>dfs.namenode.rpc-address.ns1.16</name>
<value>192.168.0.3:25000</value>
</property>
<property>
<name>dfs.namenode.rpc-address.ns1.17</name>
<value>192.168.0.4:25000</value>
</property>
<property>
<name>dfs.ha.namenodes.ns1</name>
<value>16,17</value>
</property>
<property>
<name>dfs.client.failover.proxy.provider.ns1</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
<property>
<name>dfs.client.failover.proxy.provider.hacluster</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
```


3. On FusionInsight Manager of the ClickHouse cluster, choose **Cluster** > **Services** > **ClickHouse** > **Configurations** > **All Configurations** > **ClickHouseServer(Role)** > **Engine**, and add the parameter values obtained in **Step 5.2** to the custom configuration **clickhouse-to-hdfs-customize**.

Step 6 Save the configuration and restart ClickHouse.

Step 7 Log in to the node where the client is installed as the client installation user.

Step 8 Run the following command to go to the client installation directory:

cd /opt/client

Step 9 Run the following command to configure environment variables:

source bigdata_env

Step 10 Run the following command to authenticate the current user. (Skip this step for a cluster with Kerberos authentication disabled.)

kinit clickhouseuser

Step 11 Run the client command of ClickHouse to log in to the ClickHouse client.

clickhouse client --host *Service IP address of the ClickHouseServer instance* **--secure --port 9440**

NOTE

To obtain the service IP address of the ClickHouseServer instance, perform the following steps:

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. On the page that is displayed, click the **Instances** tab. On this tab page, obtain the service IP addresses of the ClickHouseServer instance.

Step 12 Connect to the HDFS component.

- Run the following command to connect to the HDFS component in HA mode:

  CREATE TABLE default.hdfs_engine_table (`name` String, `value` UInt32) ENGINE = HDFS('hdfs://*{nameservice}***/tmp/secure_ck.txt', 'TSV')**

- Run the following command to connect to the HDFS component in non-HA mode:

  CREATE TABLE default.hdfs_engine_table (`name` String, `value` UInt32) ENGINE = HDFS('hdfs://*{namenode_ip}***:***{dfs.namenode.rpc.port}***/tmp/secure_ck.txt', 'TSV')**

NOTE

- To obtain the value of *nameservice*, perform the following steps:

  On FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** > **NameService Management** and obtain the value of **NameService**. The **nameservice** used in the table creation statement must be in lowercase (regardless of whether the value in the HDFS cluster is in lowercase). For example, if the value in the HDFS cluster is **NS1**, the value in the table creation statement must be in lowercase **ns1**.

- To obtain the value of *namenode_ip*, perform the following steps:

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS**. On the page that is displayed, click the **Instance** tab. On this tab page, obtain the service IP addresses of the active NameNode.

- To obtain the value of *dfs.namenode.rpc.port*, perform the following steps:

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS**. On the page that is displayed, click the **Configurations** tab then the **All Configurations** sub-tab. On this sub-tab page, search for **dfs.namenode.rpc.port** to obtain its value.

- HDFS file path to be accessed:

  If multiple files need to be accessed, add an asterisk (*) to the end of the folder, for example, **hdfs://***{namenode_ip}*:*{dfs.namenode.rpc.port}*/**tmp/***.

- Write data. For details, see **Process of Writing ClickHouse Data to HDFS**.

----End

Process of Writing ClickHouse Data to HDFS

To write ClickHouse data to HDFS, perform the following steps: For example, write the **secure_ck.txt** file in the **/tmp** directory of HDFS.

1. Create an HDFS table.

   CREATE TABLE *hdfs_engine_table* **(name String, value UInt32) ENGINE=HDFS('hdfs://***{namenode_ip}*:*{dfs.namenode.rpc.port}***/tmp/secure_ck.txt', 'TSV')**

2. Write an HDFS data file.

   INSERT INTO *hdfs_engine_table* **VALUES ('one', 1), ('two', 2), ('three', 3)**

3. Query the HDFS data file.

   SELECT * FROM *hdfs_engine_table* **LIMIT 2**

```
┌─name─┬─value─┐
│ one  │     1 │
│ two  │     2 │
└──────┴───────┘
```

CAUTION

- If you use ClickHouse to write data in an HDFS engine table and the target data file does not exist, it will be generated.
- ClickHouse does not support deletion, modification, and appending of HDFS engine table data.
- After the HDFS engine table is deleted from ClickHouse, data files in the HDFS remain unchanged.

3.4.5 Configuring Interconnection Between ClickHouse and Kafka

3.4.5.1 Connecting ClickHouse to the Kafka Using the Username and Password

NOTE

This topic is available for MRS 3.3.0-LTS and later versions only.

Scenario

This topic describes how to connect ClickHouse to Kafka using a username and password and consume Kafka data.

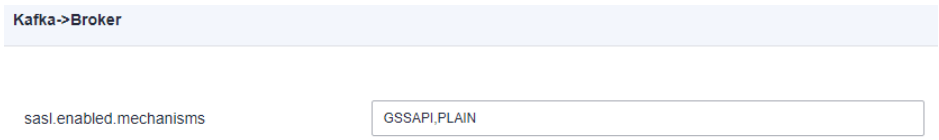
Prerequisites

- A Kafka cluster has been created and is in security mode (Kerberos authentication is enabled).
- The cluster client has been installed.
- If ClickHouse and Kafka are not in the same cluster, establish cross-cluster mutual trust between them. For details, see **Configuring Cross-Manager Mutual Trust Between Clusters**.

Procedure

Step 1 Log in to FusionInsight Manager, select **Kafka**, choose **System** > **Permission** > **User**, and click **Create User**. Create a human-machine user with Kafka permission. For example, create a human-machine user **ck_user1**. Change the initial password upon first login. For details about Kafka user permission, see **Kafka User Permissions**.

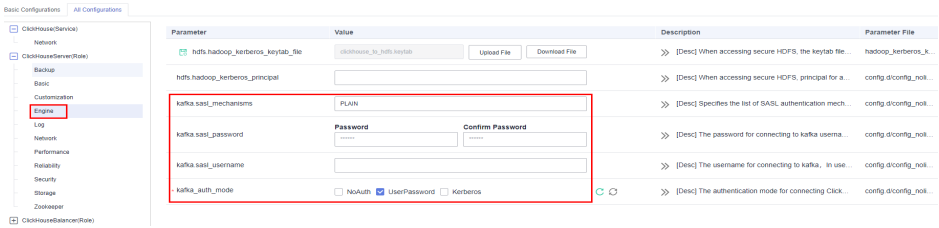
Step 2 Choose **Cluster** > **Services** > **Kafka** and choose **Configurations** > **All Configurations**. Search for **sasl.enabled.mechanisms**, and change the value to **GSSAPI,PLAIN**. Click **Save**.



Step 3 Log in to FusionInsight Manager, select **ClickHouse**, choose **Cluster** > **Services** > **ClickHouse**, and click **Configurations** > **All Configurations**. Select **ClickHouseServer (Role)** > **Engine**, and modify the parameters listed in the following table. Configure the username and password for connecting to Kafka.

Parameter	Description
kafka.sasl_mechanisms	SASL authentication for connecting to Kafka. The parameter value is **PLAIN**.

Parameter	Description
kafka.sasl_password	Password for connecting to Kafka. The initial password of the new user **ck_user1** must be changed. Otherwise, the authentication fails.
kafka.sasl_username	Username for connecting to Kafka. Enter the username created in **Step 1**.
kafka_auth_mode	Authentication mode for the ClickHouse to connect to the Kafka. Set this parameter to **UserPassword**.



Step 4 Click **Save**. In the displayed dialog box, click **OK** to save the configuration. Choose **Instances**, select **ClickHouseServer**, and click **More** > **Instance Rolling Restart**.

Step 5 Go to the Kafka client installation directory. For details, see **Using the Kafka Client**.

1. Log in to the node where the Kafka client is installed as the Kafka client installation user.
2. Run the following command to go to the client installation directory:

   cd /opt/client
3. Run the following command to configure environment variables:

   source bigdata_env
4. If Kerberos authentication is enabled for the cluster, run the following command to authenticate the user. If Kerberos authentication is disabled for the cluster, skip this step.

   kinit *Component service user*

Step 6 Run the following command to create a Kafka topic. For details, see **Creating a Kafka Topic**.

kafka-topics.sh --topic *topic1* **--create --zookeeper** *IP address of the Zookeeper role instance:Port used by ZooKeeper to listen to client*/**kafka --partitions** *2* **--replication-factor** *1*

NOTE

- **--topic**: name of the topic to be created, for example, **topic1**.
- **--zookeeper** is the IP address of the node where the ZooKeeper role instances are located, which can be the IP address of any of the three role instances. You can obtain the IP address of the node by performing the following steps:

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ZooKeeper**. On the page that is displayed, click the **Instances** tab to query the IP address of the ZooKeeper instance.
- **--partitions** and **--replication-factor** are the topic partitions and topic backup replicas, respectively. The number of the two parameters cannot exceed the number of Kafka role instances.
- To obtain the *Port used by ZooKeeper to listen to the client*, log in to FusionInsight Manager, click **Cluster**, choose **Services** > **ZooKeeper**, and view the value of **clientPort** on the **Configuration** tab page. The default port is 24002.

Step 7 Log in to the ClickHouse client node and connect it to the ClickHouse server. For details, see **ClickHouse Client Practices**.

Step 8 Create a Kafka table engine. The following is an example:

```
CREATE TABLE queue1 (
key String,
value String,
event_date DateTime
) ENGINE = Kafka()
SETTINGS kafka_broker_list = 'kafka_ip1:21007,kafka_ip2:21007,kafka_ip3:21007',
kafka_topic_list = 'topic1',
kafka_group_name = 'group1',
kafka_format = 'CSV',
kafka_row_delimiter = '\n',
kafka_handle_error_mode='stream';
```

The following table lists the related parameters.

Parameter	Description
kafka_broker_list	A list of IP addresses and port numbers of Kafka broker instances. For example, *:IP address 1 of Kafka broker instance*:9092,*IP address 2 of Kafka broker instance*:9092,*IP address 3 of Kafka broker instance*:9092 To obtain the IP address of a Kafka broker instance, perform the following operations: Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. Click the **Instances** tab to query the IP addresses of the Kafka instances.
kafka_topic_list	Topic where Kafka data is consumed
kafka_group_name	Kafka consumer group
kafka_format	Formatting type of consumed data. **JSONEachRow** indicates the JSON format (a piece of data in each line). **CSV** indicates the data is in a line but separated by commas (,).
kafka_row_delimiter	Delimiter character, which ends a message.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>kafka_handle_error_mode</td><td>If this parameter is set to stream, each message processing exception is printed. You need to create a view and query the specific exception of abnormal data through the view.

The following example shows you how to create a view:
<pre>
CREATE MATERIALIZED VIEW default.kafka_errors2
(
`topic` String,
`key` String,
`partition` Int64,
`offset` Int64,
`timestamp` Date,
`timestamp_ms` Int64,
`raw` String,
`error` String
)
ENGINE = MergeTree
ORDER BY (topic, partition, offset)
SETTINGS index_granularity = 8192 AS
SELECT
_topic AS topic,
_key AS key,
_partition AS partition,
_offset AS offset,
_timestamp AS timestamp,
_timestamp_ms AS timestamp_ms,
_raw_message AS raw,
_error AS error
FROM default.queue1;
</pre>
Query the view. The following is an example:
<pre>
host1 :) select * from kafka_errors2; SELECT * FROM kafka_errors2
Query id: bf4d788f-bcb9-44f5-95d0-a6c83c591ddb
┌─topic───┬─key─┬─partition─┬─offset─┬───timestamp─┬─timestam
p_ms─┬─raw─┬─error──
──
──┐ │ topic1
│   │   1 │   8 │ 2023-06-20 │   1687252213 │ 456 │ Cannot
parse date: value is too short: (at row 1) Buffer has gone, cannot extract
information about what has been parsed. │
└─────────┴───────┴───────────────┴──────────┴────────────
───┴────────────────┴───────┴─────────────────────────────
──
──
──────┘ 1 rows in set. Elapsed: 0.003 sec.   host1 :)
</pre></td></tr>
<tr><td>kafka_skip_broken_messages</td><td>(Optional) Number of Kafka data records where parsing exceptions are ignored. If N exceptions occur and the background thread ends, the materialized view is re-arranged to monitor the data.</td></tr>
<tr><td>kafka_num_consumers</td><td>(Optional) Number of consumers of a single Kafka engine. You can set this parameter to a larger value to improve the consumption data throughput. But the maximum value of this parameter cannot exceed the total number of partitions of the corresponding topic.</td></tr>
</table>

For details about other configurations, see **https://clickhouse.com/docs/en/engines/table-engines/integrations/kafka**.

Step 9 Connect the client to ClickHouse to create a local table. The following is an example:

```
CREATE TABLE daily1(
key String,
value String,
event_date DateTime
)ENGINE = MergeTree()
ORDER BY key;
```

Step 10 Connect the client to ClickHouse to create a materialized view. The following is an example:

```
CREATE MATERIALIZED VIEW default.consumer TO default.daily1 (
`event_date` DateTime,
`key` String,
`value` String
) AS
SELECT
event_date,
key,
value
FROM default.queue1;
```

Step 11 Perform **Step 5** again to go to the Kafka client installation directory.

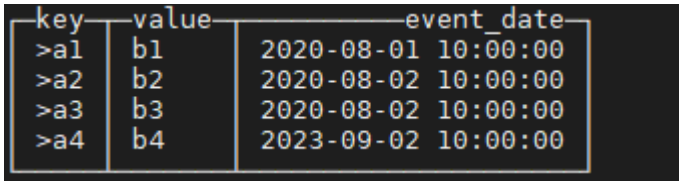
Step 12 Run the following command to send a message to the topic created in **Step 6**:

kafka-console-producer.sh --broker-list *IP address 1 of the Kafka broker instance*:**9092,***IP address 2 of the Kafka broker instance*:**9092,***IP address 3 of the Kafka broker instance*:**9092 --topic** *topic1*

```
>a1,b1,'2020-08-01 10:00:00'
>a2,b2,'2020-08-02 10:00:00'
>a3,b3,'2020-08-02 10:00:00'
>a4,b4,'2023-09-02 10:00:00'
```

Step 13 Query the consumed Kafka data and the preceding materialized view. The following is an example:

```
select * from daily1;
```



----End

3.4.5.2 Interconnecting ClickHouse with Kafka Through Kerberos Authentication

NOTE

This topic is available for MRS 3.3.0-LTS and later versions only.

Scenario

This topic describes how to connect ClickHouse to Kafka using Kerberos authentication and how to consume Kafka data.

Prerequisites

- A Kafka cluster has been created and is in security mode (Kerberos authentication is enabled).
- The cluster client has been installed.
- If ClickHouse and Kafka are not in the same cluster, establish cross-cluster mutual trust between them. For details, see **Configuring Cross-Manager Mutual Trust Between Clusters**.

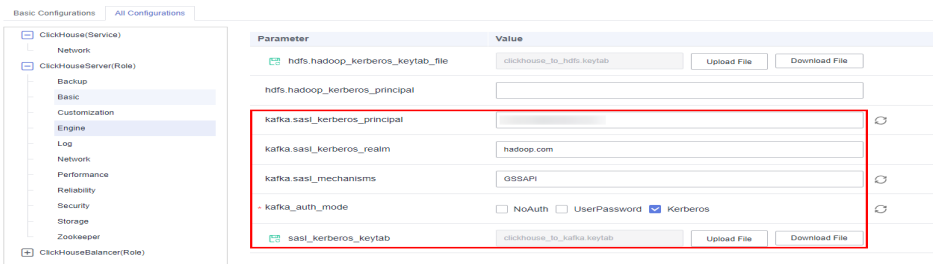
Procedure

Step 1 Log in to the FusionInsight Manager of the cluster where Kafka is deployed, choose **System** > **Permission** > **User** > **Create User**, and create a human-machine user with the Kafka permission. For example, create a human-machine user **ck_user1**. For details about Kafka user permission, see **Kafka User Permissions**.

Step 2 Choose **System** > **Permission** > **User**. On the displayed page, locate the **ck_user1** user, and click **More** > **Download Authentication Credential** in the **Operation** column of the user. Save the file and decompress it to obtain the **user.keytab** and **krb5.conf** files. Rename the **user.keytab** file **clickhouse_to_kafka.keytab**.

Step 3 Log in to the FusionInsight Manager of the cluster where ClickHouse is deployed, and choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** then **All Configurations**, and click **ClickHouseServer(Role)** > **Engine**. The following table shows the parameter needs to be configured.

Parameter	Description
kafka.sasl_kerberos_principal	Principal for connecting to Kafka. Enter the username created in **Step 1**.
kafka.sasl_kerberos_realm	Domain name of the Kafka cluster
kafka.sasl_mechanisms	SASL authentication for connecting to Kafka. The parameter value is **GSSAPI**.
kafka_auth_mode	Authentication mode for the ClickHouse to connect to the Kafka. Set this parameter to **Kerberos**.
sasl_kerberos_keytab	Authentication file for connecting to Kafka, which is the **clickhouse_to_kafka.keytab** file uploaded in **Step 2**.



Step 4 Click **Save**. In the displayed dialog box, click **OK** to save the configuration. Choose **Instances**, select **ClickHouseServer**, and click **More** > **Instance Rolling Restart**.

Step 5 Go to the Kafka client installation directory. For details, see **Using the Kafka Client**.

1. Log in to the node where the Kafka client is installed as the Kafka client installation user.
2. Run the following command to go to the client installation directory:

   cd /opt/client
3. Run the following command to configure environment variables:

   source bigdata_env
4. Run the following command to authenticate the user:

   kinit *Component service user*

Step 6 Run the following command to create a Kafka topic. For details, see **Creating a Kafka Topic**.

kafka-topics.sh --topic *topic1* **--create --zookeeper** *IP address of the Zookeeper role instance:Port used by ZooKeeper to listen to client*/**kafka --partitions** *2* **--replication-factor** *1*

NOTE

- **--topic**: name of the topic to be created, for example, **topic1**.
- **--zookeeper** is the IP address of the node where the ZooKeeper role instances are located, which can be the IP address of any of the three role instances. You can obtain the IP address of the node by performing the following steps:

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ZooKeeper**. On the page that is displayed, click the **Instances** tab to query the IP address of the ZooKeeper instance.
- **--partitions** and **--replication-factor** are the topic partitions and topic backup replicas, respectively. The number of the two parameters cannot exceed the number of Kafka role instances.
- To obtain the *Port used by ZooKeeper to listen to the client*, log in to FusionInsight Manager, click **Cluster**, choose **Services** > **ZooKeeper**, and view the value of **clientPort** on the **Configuration** tab page. The default port is 24002.

Step 7 Log in to the ClickHouse client node and connect it to the ClickHouse server. For details, see **ClickHouse Client Practices**.

Step 8 Create a Kafka table engine. The following is an example:

```
CREATE TABLE queue1 (
key String,
value String,
event_date DateTime
) ENGINE = Kafka()
SETTINGS kafka_broker_list = 'kafka_ip1:21007,kafka_ip2:21007,kafka_ip3:21007',
kafka_topic_list = 'topic1',
kafka_group_name = 'group2',
kafka_format = 'CSV',
kafka_row_delimiter = '\n',
kafka_handle_error_mode='stream';
```

The following table lists the related parameters.

Parameter	Description
kafka_broker_list	A list of IP addresses and port numbers of Kafka broker instances. For example, *:IP address 1 of Kafka broker instance*:9092,*IP address 2 of Kafka broker instance*:9092,*IP address 3 of Kafka broker instance*:9092 To obtain the IP address of a Kafka broker instance, perform the following operations: Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. Click the **Instance** tab to query the IP addresses of the Kafka instances.
kafka_topic_list	Topic where Kafka data is consumed
kafka_group_name	Kafka consumer group
kafka_format	Formatting type of consumed data. **JSONEachRow** indicates the JSON format (a piece of data in each line). **CSV** indicates the data is in a line but separated by commas (,).
kafka_row_delimiter	Delimiter character, which ends a message.

Parameter	Description

kafka_handle_error_mode	If this parameter is set to **stream**, each message processing exception is printed. You need to create a view and query the specific exception of abnormal data through the view.
	The following example shows you how to create a view:
	`CREATE MATERIALIZED VIEW default.kafka_errors2`
	`(`
	`` `topic` String, ``
	`` `key` String, ``
	`` `partition` Int64, ``
	`` `offset` Int64, ``
	`` `timestamp` Date, ``
	`` `timestamp_ms` Int64, ``
	`` `raw` String, ``
	`` `error` String ``
	`)`
	`ENGINE = MergeTree`
	`ORDER BY (topic, partition, offset)`
	`SETTINGS index_granularity = 8192 AS`
	`SELECT`
	`_topic AS topic,`
	`_key  AS key,`
	`_partition AS partition,`
	`_offset AS offset,`
	`_timestamp AS timestamp,`
	`_timestamp_ms AS timestamp_ms,`
	`_raw_message AS raw,`
	`_error AS error`
	`FROM default.queue1;`
	Query the view. The following is an example:
	`host1 :) select * from kafka_errors2;  SELECT * FROM kafka_errors2`
	`Query id: bf4d788f-bcb9-44f5-95d0-a6c83c591ddb`
	`┌─topic──┬─key─┬─partition─┬─offset─┬──timestamp─┬─timestamp_ms─┬─raw─┬─error─────────────┐`
	`│ topic1 │     │         1 │      8 │ 2023-06-20 │   1687252213 │ 456 │ Cannot parse date: value is too short: (at row 1) Buffer has gone, cannot extract information about what has been parsed. │`
	`└────────┴─────┴───────────┴────────┴────────────┴──────────────┴─────┴───────────────────┘`
	`1 rows in set. Elapsed: 0.003 sec.   host1 :)`

kafka_skip_broken_messages	(Optional) Number of Kafka data records where parsing exceptions are ignored. If *N* exceptions occur and the background thread ends, the materialized view is re-arranged to monitor the data.
kafka_num_consumers	(Optional) Number of consumers of a single Kafka engine. You can set this parameter to a larger value to improve the data consumption throughput. But the maximum value of this parameter cannot exceed the total number of partitions of the corresponding topic.

For details about other configurations, see **https://clickhouse.com/docs/en/engines/table-engines/integrations/kafka**.

Step 9 Connect the client to ClickHouse to create a local table. The following is an example:

```
CREATE TABLE daily1(
key String,
value String,
event_date DateTime
)ENGINE = MergeTree()
ORDER BY key;
```

Step 10 Connect the client to ClickHouse to create a materialized view. The following is an example:

```
CREATE MATERIALIZED VIEW default.consumer1 TO default.daily1 (
`event_date` DateTime,
`key` String,
`value` String
) AS
SELECT
event_date,
key,
value
FROM default.queue1;
```

Step 11 Perform **Step 5** again to go to the Kafka client installation directory.

Step 12 Run the following command to send a message to the topic created in **Step 6**:

kafka-console-producer.sh --broker-list *IP address 1 of the Kafka broker instance*:**9092,***IP address 2 of the Kafka broker instance*:**9092,***IP address 3 of the Kafka broker instance*:**9092 --topic** *topic1*

```
>a1,b1,'2020-08-01 10:00:00'
>a2,b2,'2020-08-02 10:00:00'
>a3,b3,'2020-08-02 10:00:00'
>a4,b4,'2023-09-02 10:00:00'
```

Step 13 Query the consumed Kafka data and the preceding materialized view. The following is an example:

```
select * from daily1;
```

----End

3.4.5.3 Connecting ClickHouse to Kafka in Normal Mode

NOTE

This topic is available for MRS 3.3.0-LTS and later versions only.

Scenario

This topic describes how to connect to Kafka in normal mode and consume Kafka data.

Prerequisites

- A Kafka cluster has been created and is in normal mode (Kerberos authentication is disabled).
- You have created a ClickHouse cluster and installed the ClickHouse client. The ClickHouse and Kafka clusters can communicate with each other.

Procedure

Step 1 Log in to the FusionInsight Manager of the cluster where ClickHouse is deployed, and choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** then **All Configurations**, and click **ClickHouseServer(Role)** > **Engine**. The following table shows the parameter needs to be configured.

Parameter	Description
kafka_auth_mode	Authentication method for the connection between ClickHouse and Kafka. Set this parameter to **NoAuth**.

Step 2 Choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** > **All Configurations** > **ClickHouseServer (Role)** > **Customization**, and add the following parameters to **clickhouse-config-customize**.

Name	Value
kafka.security_protocol	plaintext

Step 3 Click **Save**. In the displayed dialog box, click **OK** to save the configuration. Choose **Instances**, select **ClickHouseServer**, and click **More** > **Instance Rolling Restart**.

Step 4 Go to the Kafka client installation directory. For details, see **Using the Kafka Client**.

1. Log in to the node where the Kafka client is installed as the Kafka client installation user.
2. Run the following command to go to the client installation directory:

   cd /opt/client
3. Run the following command to configure environment variables:

   source bigdata_env
4. If Kerberos authentication is enabled for the cluster, run the following command to authenticate the user. If Kerberos authentication is disabled for the cluster, skip this step.

   kinit *Component service user*

Step 5 Run the following command to create a Kafka topic. For details, see **Creating a Kafka Topic**.

kafka-topics.sh --topic *topic1* **--create --zookeeper** *IP address of the Zookeeper role instance:Port used by ZooKeeper to listen to client*/**kafka --partitions** *2* **--replication-factor** *1*

NOTE

- **--topic**: name of the topic to be created, for example, **topic1**.
- **--zookeeper** is the IP address of the node where the ZooKeeper role instances are located, which can be the IP address of any of the three role instances. You can obtain the IP address of the node by performing the following steps:

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ZooKeeper**. On the page that is displayed, click the **Instances** tab to query the IP address of the ZooKeeper instance.
- **--partitions** and **--replication-factor** are the topic partitions and topic backup replicas, respectively. The number of the two parameters cannot exceed the number of Kafka role instances.
- To obtain the *Port used by ZooKeeper to listen to the client*, log in to FusionInsight Manager, click **Cluster**, choose **Services** > **ZooKeeper**, and view the value of **clientPort** on the **Configuration** tab page. The default port is 24002.

Step 6 Log in to the ClickHouse client node and connect it to the ClickHouse server. For details, see **ClickHouse Client Practices**.

Step 7 Create a Kafka table engine. The following is an example:

```
CREATE TABLE queue1 (
key String,
value String,
event_date DateTime
) ENGINE = Kafka()
SETTINGS kafka_broker_list = 'kafka_ip1:21005,kafka_ip2:21005,kafka_ip3:21005',
kafka_topic_list = 'topic1',
kafka_group_name = 'group2',
kafka_format = 'CSV',
kafka_row_delimiter = '\n',
kafka_handle_error_mode='stream';
```

The following table lists the related parameters.

Parameter	Description
kafka_broker_list	A list of IP addresses and port numbers of Kafka broker instances. For example, *:IP address 1 of Kafka broker instance*:9092,*IP address 2 of Kafka broker instance*:9092,*IP address 3 of Kafka broker instance*:9092 To obtain the IP address of the Kafka broker instance, perform the following steps: Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. Click the **Instance** tab to query the IP addresses of the Kafka instances.
kafka_topic_list	Topic where Kafka data is consumed
kafka_group_name	Kafka consumer group

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>kafka_format</td><td>Formatting type of consumed data. JSONEachRow indicates the JSON format (a piece of data in each line). CSV indicates the data is in a line but separated by commas (,). For more information, visit https://clickhouse.tech/docs/en/interfaces/formats/.</td></tr>
<tr><td>kafka_row_delimiter</td><td>Delimiter character, which ends a message.</td></tr>
<tr><td>kafka_handle_error_mode</td><td>If this parameter is set to stream, each message processing exception is printed. You need to create a view and query the specific exception of abnormal data through the view.

The following example shows you how to create a view:
CREATE MATERIALIZED VIEW default.kafka_errors2
(
`topic` String,
`key` String,
`partition` Int64,
`offset` Int64,
`timestamp` Date,
`timestamp_ms` Int64,
`raw` String,
`error` String
)
ENGINE = MergeTree
ORDER BY (topic, partition, offset)
SETTINGS index_granularity = 8192 AS
SELECT
_topic AS topic,
_key AS key,
_partition AS partition,
_offset AS offset,
_timestamp AS timestamp,
_timestamp_ms AS timestamp_ms,
_raw_message AS raw,
_error AS error
FROM default.queue1;

Query the view. The following is an example:
host1 :) select * from kafka_errors2; SELECT * FROM kafka_errors2
Query id: bf4d788f-bcb9-44f5-95d0-a6c83c591ddb
┌─topic──┬─key─┬─partition─┬─offset─┬──timestamp─┬─timestam
p_ms─┬─raw─┬─error───────────────────────────
──
──────────────────────────────┐ │ topic1
│ │ 1 │ 8 │ 2023-06-20 │ 1687252213 │ 456 │ Cannot
parse date: value is too short: (at row 1) Buffer has gone, cannot extract
information about what has been parsed. │
└──────┴──────┴──────┴──────────┴──────────
──┴──────────┴──────┴──────────────────
──
──
─────┘ 1 rows in set. Elapsed: 0.003 sec.   host1 :)</td></tr>
<tr><td>kafka_skip_broken_messages</td><td>(Optional) Number of Kafka data records where parsing exceptions are ignored. If N exceptions occur and the background thread ends, the materialized view is re-arranged to monitor the data.</td></tr>
</table>

Parameter	Description
kafka_num_consumers	(Optional) Number of consumers of a single Kafka engine. You can set this parameter to a larger value to improve the data consumption throughput. But the maximum value of this parameter cannot exceed the total number of partitions of the corresponding topic.

For details about other configurations, see **https://clickhouse.com/docs/en/engines/table-engines/integrations/kafka**.

Step 8 Connect the client to ClickHouse to create a local table. The following is an example:

```
CREATE TABLE daily1(
key String,
value String,
event_date DateTime
)ENGINE = MergeTree()
ORDER BY key;
```

Step 9 Connect the client to ClickHouse to create a materialized view. The following is an example:

```
CREATE MATERIALIZED VIEW default.consumer1 TO default.daily1 (
`event_date` DateTime,
`key` String,
`value` String
) AS
SELECT
event_date,
key,
value
FROM default.queue1;
```

Step 10 Perform **Step 4** again to go to the Kafka client installation directory.

Step 11 Run the following command to send a message to the topic created in **Step 5**:

kafka-console-producer.sh --broker-list *IP address 1 of the Kafka broker instance*:**9092,***IP address 2 of the Kafka broker instance*:**9092,***IP address 3 of the Kafka broker instance*:**9092 --topic** *topic1*

```
>a1,b1,'2020-08-01 10:00:00'
>a2,b2,'2020-08-02 10:00:00'
>a3,b3,'2020-08-02 10:00:00'
>a4,b4,'2023-09-02 10:00:00'
```

Step 12 Query the consumed Kafka data and the preceding materialized view. The following is an example:

```
select * from daily;
```

```
┌─key─┬─value─┬──────────event_date─┐
│ >a1 │ b1    │ 2020-08-01 10:00:00 │
│ >a2 │ b2    │ 2020-08-02 10:00:00 │
│ >a3 │ b3    │ 2020-08-02 10:00:00 │
│ >a4 │ b4    │ 2023-09-02 10:00:00 │
└─────┴───────┴─────────────────────┘
```

----End

3.4.6 Synchronizing Kafka Data to ClickHouse

This topic describes how to create a Kafka engine table to automatically synchronize Kafka data to the ClickHouse cluster.

Prerequisites

- You have created a Kafka cluster. The Kafka client has been installed.For details, see **Installing a Client**.
- You have created a ClickHouse cluster and installed the ClickHouse client. The ClickHouse and Kafka clusters are in the same VPC and can communicate with each other.

Constraints

Currently, ClickHouse cannot interconnect with Kafka clusters with security mode enabled.

Syntax of the Kafka Table

- **Syntax**

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = Kafka()
SETTINGS
    kafka_broker_list = 'host1:port1,host2:port2',
    kafka_topic_list = 'topic1,topic2,...',
    kafka_group_name = 'group_name',
    kafka_format = 'data_format';
    [kafka_row_delimiter = 'delimiter_symbol',]
    [kafka_schema = '',]
    [kafka_num_consumers = N]
```

- **Parameter description**

Table 3-6 Kafka table parameters

Parameter	Mandatory	Description
kafka_broker_list	Yes	A list of Kafka broker instances, separated by comma (,). For example, *IP address 1 of the Kafka broker instance*:**9092**,*IP address 2 of the Kafka broker instance*:**9092**,*IP address 3 of the Kafka broker instance*:**9092**. **NOTE** If the Kerberos authentication is enabled, parameter **allow.everyone.if.no.acl.found** must be set to **true** if port **21005** is used. Otherwise, an error will be reported. To obtain the IP address of the Kafka broker instance, perform the following steps: Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. Click the **Instances** tab to query the IP addresses of the Kafka instances.
kafka_topic_list	Yes	A list of Kafka topics.
kafka_group_name	Yes	A group of Kafka consumers, which can be customized.
kafka_format	Yes	Kafka message format, for example, JSONEachRow, CSV, and XML.
kafka_row_delimiter	No	Delimiter character, which ends a message.
kafka_schema	No	Parameter that must be used if the format requires a schema definition.
kafka_num_consumers	No	Number of consumers in per table. The default value is **1**. If the throughput of a consumer is insufficient, more consumers are required. The total number of consumers cannot exceed the number of partitions in a topic because only one consumer can be allocated to each partition.

How to Synchronize Kafka Data to ClickHouse

Step 1 Switch to the Kafka client installation directory. For details, see **Using the Kafka Client**.

1. Log in to the node where the Kafka client is installed as the Kafka client installation user.
2. Run the following command to go to the client installation directory:

   cd /opt/client
3. Run the following command to configure environment variables:

source bigdata_env

4. If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the current user. If Kerberos authentication is disabled for the current cluster, skip this step.

   kinit *Component service user*

Step 2 Run the following command to create a Kafka topic. For details, see **Creating a Kafka Topic**.

kafka-topics.sh --topic *kafkacktest2* **--create --zookeeper** *IP address of the Zookeeper role instance:Port used by ZooKeeper to listen to client*/**kafka --partitions** *2* **--replication-factor** *1*

NOTE

- **--topic** is the name of the topic to be created, for example, **kafkacktest2**.
- **--zookeeper** is the IP address of the node where the ZooKeeper role instances are located, which can be the IP address of any of the three role instances. You can obtain the IP address of the node by performing the following steps:

  Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **ZooKeeper** > **Instances**. View the IP addresses of the ZooKeeper role instance.
- **--partitions** and **--replication-factor** are the topic partitions and topic backup replicas, respectively. The number of the two parameters cannot exceed the number of Kafka role instances.
- To obtain the *Port used by ZooKeeper to listen to client*, log in to FusionInsight Manager, click **Cluster**, choose **Services** > **ZooKeeper**, and view the value of **clientPort** on the **Configuration** tab page. The default value is **24002**.

Step 3 Log in to the ClickHouse client by referring to **ClickHouse Client Practices**.

1. Run the following command to go to the client installation directory:

   cd /opt/client

2. Run the following command to configure environment variables:

   source bigdata_env

3. If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the current user. The user must have the permission to create ClickHouse tables. Therefore, you need to bind the corresponding role to the user. For details, see **Creating a ClickHouse Role**. If Kerberos authentication is disabled for the current cluster, skip this step.

   kinit *Component service user*

   Example: **kinit clickhouseuser**

4. Run the following command to connect to the ClickHouse instance node to which data is to be imported:

   clickhouse client --host *IP address of the ClickHouse instance* **--user** *Login username* **--password --port** *ClickHouse port number* **--database** *Database name* **--multiline**

   Enter the user password.

Step 4 Create a Kafka table in ClickHouse by referring to **Syntax of the Kafka Table**. For example, the following table creation statement is used to create a Kafka table whose name is **kafka_src_tbl3**, topic name is **kafkacktest2**, and message format is **JSONEachRow** in the default database.


```
create table kafka_src_tbl3 on cluster default_cluster
(id UInt32, age UInt32, msg String)
ENGINE=Kafka()
SETTINGS
 kafka_broker_list='IP address 1 of the Kafka broker instance:9092,IP address 2 of the Kafka broker instance:9092,IP address 3 of the Kafka broker instance:9092',
 kafka_topic_list='kafkacktest2',
 kafka_group_name='cg12',
 kafka_format='JSONEachRow';
```

Step 5 Create a ClickHouse replicated table, for example, the ReplicatedMergeTree table named **kafka_dest_tbl3**.

```
create table kafka_dest_tbl3 on cluster default_cluster
( id UInt32, age UInt32, msg String )
engine = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/kafka_dest_tbl3', '{replica}')
partition by age
order by id;
```

Step 6 Create a materialized view, which converts data in Kafka in the background and saves the data to the created ClickHouse table.

```
create materialized view consumer3 on cluster default_cluster to kafka_dest_tbl3 as select * from kafka_src_tbl3;
```

Step 7 Perform **Step 1** again to go to the Kafka client installation directory.

Step 8 Run the following command to send a message to the topic created in **Step 2**:

kafka-console-producer.sh --broker-list *IP address 1 of the kafka broker instance*:**9092,***IP address 2 of the kafka broker instance*:**9092,***IP address 3 of the kafka broker instance*:**9092 --topic** *kafkacktest2*

```
>{"id":31, "age":30, "msg":"31 years old"}
>{"id":32, "age":30, "msg":"31 years old"}
>{"id":33, "age":30, "msg":"31 years old"}
>{"id":35, "age":30, "msg":"31 years old"}
```

Step 9 Use the ClickHouse client to log in to the ClickHouse instance node in **Step 3** and query the ClickHouse table data, for example, to query the replicated table **kafka_dest_tbl3**. It shows that the data in the Kafka message has been synchronized to this table.

```
select * from kafka_dest_tbl3;
```

----End

3.4.7 Importing DWS Table Data to ClickHouse

ClickHouse supports the import and export of files in CSV or JSON format. This section describes how to export table data from the Data Warehouse Service (DWS) to a CSV file and then import the CSV file to a ClickHouse table.

Prerequisites

- The ClickHouse cluster and instances are normal.
- You have created a DWS cluster and obtained the username and password of the database where the related table is located.
- You have installed the MRS client, for example, in the **/opt/client** directory. The client directory in the following operations is only an example. Change it to the actual installation directory. Before using the client, download and update the client configuration file, and ensure that the active management node of Manager is available.

Importing DWS Service Data to ClickHouse

Step 1 Download Data Studio. For details, see "Data Studio GUI Client" in **Downloading the Related Tools**.

Step 2 Connect to a DWS database using the username and password of the database in the DWS cluster. For details, see **Adding a Connection**.

Step 3 Export table data from the DWS database to a CSV file.

1. (Optional) If a table and data already exist in the DWS database, skip this step. The following demonstrates how to create a test table in DWS and insert test data.

   Use Data Studio to create a test table **warehouse_t1** and insert test data.

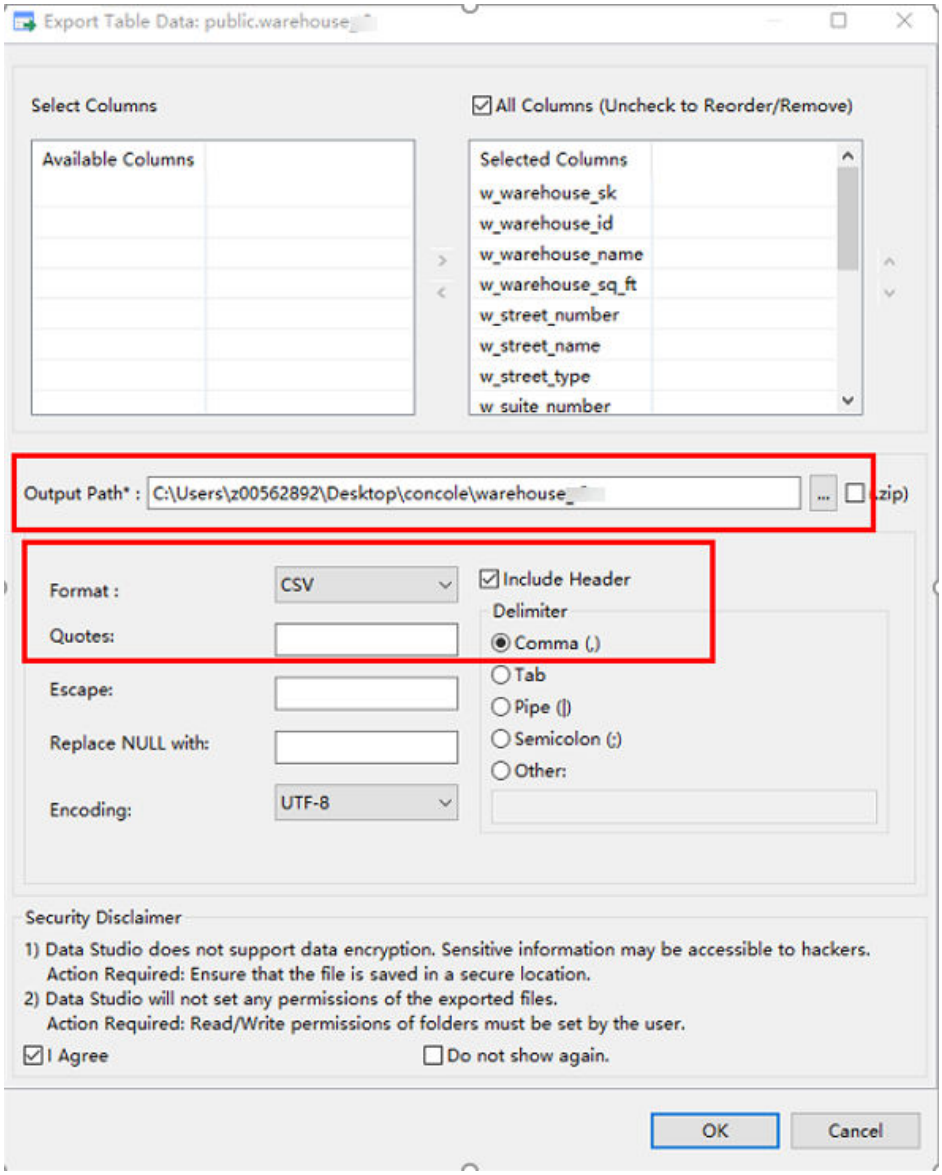
```
CREATE TABLE warehouse_t1
 (
  W_WAREHOUSE_SK INTEGER NOT NULL,
  W_WAREHOUSE_ID CHAR ( 16 ) NOT NULL,
  W_WAREHOUSE_NAME VARCHAR ( 20 ),
  W_WAREHOUSE_SQ_FT INTEGER,
  W_STREET_NUMBER CHAR ( 10 ),
  W_STREET_NAME VARCHAR ( 60 ),
  W_STREET_TYPE CHAR ( 15 ),
  W_SUITE_NUMBER CHAR ( 10 ),
  W_CITY VARCHAR ( 60 ),
  W_COUNTY VARCHAR ( 30 ),
  W_STATE CHAR ( 2 ),
  W_ZIP CHAR ( 10 ),
  W_COUNTRY VARCHAR ( 20 ),
  W_GMT_OFFSET DECIMAL ( 5,2 ),
  W_DATE DATE
 );

INSERT INTO warehouse_t1 VALUES(1314, 123, 'name1', 2324, 123, 'STREET_NAME1', '12', '12',
'guangzhou', 'zhongguo', '1', '12', 'zn', 50.2, '2021-07-05 17:45:07');
INSERT INTO warehouse_t1 VALUES(1314, 123, 'name2', 2324, 123, 'STREET_NAME2', '12', '12',
'guangzhou', 'zhongguo', '1', '12', 'zn', 50.2, '2021-07-05 17:45:08');
INSERT INTO warehouse_t1 VALUES(1314, 123, 'name3', 2324, 123, 'STREET_NAME3', '12', '12',
'guangzhou', 'zhongguo', '1', '12', 'zn', 50.2, '2021-07-05 17:45:09');
INSERT INTO warehouse_t1 VALUES(1314, 123, 'name4', 2324, 123, 'STREET_NAME4', '12', '12',
'guangzhou', 'zhongguo', '1', '12', 'zn', 50.2, '2021-07-05 17:45:00');
INSERT INTO warehouse_t1 VALUES(1314, 123, 'name5', 2324, 123, 'STREET_NAME5', '12', '12',
'guangzhou', 'zhongguo', '1', '12', 'zn', 50.2, '2021-07-05 17:45:01');
INSERT INTO warehouse_t1 VALUES(1314, 123, 'name6', 2324, 123, 'STREET_NAME6', '12', '12',
```



```
'guangzhou', 'zhongguo', '1', '12', 'zn', 50.2, '2021-07-05 17:45:02');
INSERT INTO warehouse_t1 VALUES(1314, 123, 'name7', 2324, 123, 'STREET_NAME7', '12', '12',
'guangzhou', 'zhongguo', '1', '12', 'zn', 50.2, '2021-07-05 17:45:03');
INSERT INTO warehouse_t1 VALUES(1314, 123, 'name8', 2324, 123, 'STREET_NAME8', '12', '12',
'guangzhou', 'zhongguo', '1', '12', 'zn', 50.2, '2021-07-05 17:45:04');
INSERT INTO warehouse_t1 VALUES(1314, 123, 'name9', 2324, 123, 'STREET_NAME9', '12', '12',
'guangzhou', 'zhongguo', '1', '12', 'zn', 50.2, '2021-07-05 17:45:05');
INSERT INTO warehouse_t1 VALUES(1314, 123, 'name0', 2324, 123, 'STREET_NAME0', '12', '12',
'guangzhou', 'zhongguo', '1', '12', 'zn', 50.2, '2021-07-05 17:45:06');
INSERT INTO warehouse_t1(W_WAREHOUSE_SK, W_WAREHOUSE_ID, W_WAREHOUSE_NAME,
W_DATE) VALUES(1314, 123, 'name0', '2021-07-05 17:45:06');
```

2. Export the DWS table data to a CSV file.

   In **Object Browser** on the left of Data Studio, right-click the table to be exported and select **Export Table Data**. On the export page, select a specific output path, set **Format** to **CSV** and **Delimiter** to **Comma (,)**, select **I Agree** under **Security Disclaimer**, and click **OK**. In this example, the data file **warehouse_t1.csv** of the **warehouse_t1** table is exported.



Step 4 Use WinSCP to upload the exported CSV file to the directory of the ClickHouse instance node. For example, upload the **warehouse_t1.csv** file to the **/opt** directory.

Step 5 Log in to the node where the ClickHouse client is installed as the client installation user.

Step 6 Run the following command to go to the client installation directory:

cd /opt/client

Step 7 Run the following command to configure environment variables:

source bigdata_env

Step 8 If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the current user. The user must have the permission to create ClickHouse tables. Therefore, you need to bind the corresponding role to the user. For details, see **Creating a ClickHouse Role**. If Kerberos authentication is disabled for the current cluster, skip this step.

1. Run the following command if it is an MRS 3.1.0 cluster:

   export CLICKHOUSE_SECURITY_ENABLED=true

2. **kinit** *Component service user*

   Example: **kinit clickhouseuser**

Step 9 Run the following command to connect to the ClickHouse instance node to which data is to be imported:

clickhouse client --host *IP address of the ClickHouse instance* **--user** *Login username* **--password --port** *ClickHouse port number* **--database** *Database name*

Enter the user password.

Step 10 Create a table with the same structure as the DWS table on the ClickHouse instance node.

For example, run the following table creation statements to create a ReplicatedMergeTree table **warehouse_t1** with the same table structure as that in **Step 3** under the default database and user on the ClickHouse instance.

```
CREATE TABLE warehouse_t1
(
    `W_WAREHOUSE_SK` Int32 NOT NULL,
    `W_WAREHOUSE_ID` String NOT NULL,
    `W_WAREHOUSE_NAME` String,
    `W_WAREHOUSE_SQ_FT` Int32,
    `W_STREET_NUMBER` String,
    `W_STREET_NAME` String,
    `W_STREET_TYPE` String,
    `W_SUITE_NUMBER` String,
    `W_CITY` String,
    `W_COUNTY` String,
    `W_STATE` String,
    `W_ZIP` String,
    `W_COUNTRY` String,
    `W_GMT_OFFSET` Decimal(5, 2),
    `W_DATE` DateTime
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/warehouse_t1', '{replica}')
PARTITION BY toYear(W_DATE)
ORDER BY (W_DATE, W_WAREHOUSE_ID);
```


Step 11 Exit the ClickHouse client.

exit;

Step 12 Run the following command to import the data in the exported CSV file to the ClickHouse table:

clickhouse client --host *ClickHouse instance IP address* **--database** *Database name* **--port** *Port number* **--format_csv_delimiter="***CSV file delimiter***" --query="INSERT INTO** *Table name* **FORMAT CSV" <** *Host path where the CSV file is stored*

For example, import the comma-separated CSV file **warehouse_t1.csv** to the **warehouse_t1** table under the default database and user.

clickhouse client --host 10.248.12.10 --format_csv_delimiter="," --query="INSERT INTO warehouse_t1 FORMAT CSV" < /opt/warehouse_t1.csv

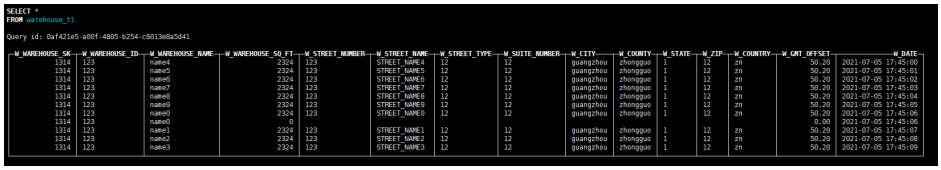
Step 13 After the import is complete, log in to the ClickHouse client, connect to the ClickHouse instance node where the data is imported, and run the query command to check the import result.

For example, query data in the **warehouse_t1** table after the import is complete. The result is shown in the following figure.

clickhouse client --host *ClickHouse instance IP address* **--user** *Login username* **--password** *Password* **--port** *ClickHouse port number* **--database** *Database name*

Enter the user password.

select * from warehouse_t1;



----End

3.4.8 Importing ClickHouse Data in Batches

NOTE

This topic is available in MRS 3.3.0 or later only.

Scenario

If a large number of data files need to be imported, you can use the multi-thread import tool to import ClickHouse data files in batches.

Prerequisites

- The ClickHouse client has been installed in a directory, for example, **/opt/client**.
- For a cluster in security mode, a user with ClickHouse permissions has been created, for example, **clickhouseuser**. For details, see **Creating a ClickHouse Role**.

- The data file to be imported has been uploaded to a client node directory, for example, **/opt/data**. For details about all data types supported by ClickHouse, visit **https://clickhouse.com/docs/en/interfaces/formats**.

Procedure

Step 1 Log in to the node where the client is installed as the client installation user.

Step 2 Go to the directory where the multi-thread write tool **clickhouse_insert_tool** is deployed.

cd /opt/client/ClickHouse/clickhouse_insert_tool

Step 3 Use the text editor to open **clickhouse_insert_tool.sh** and enter required information based on the comments.

Parameter	Description	Example
datapath	Directory containing the data to be imported	/opt/data
balancer_ip_list	IP addresses of the ClickHouse Balancer instances. The IP addresses must be enclosed in parentheses. A single IP address must be enclosed in double quotation marks, and IP addresses must be separated by spaces.	("192.168.1.1" "192.168.1.2")
balancer_tcp_port	TCP port for the Balancer instance of the ClickHouse service	21428
local_table_name	Names of the local library and table to be imported	testdb1.testtb1
thread_num	Number of concurrent threads for importing data	10
data_format	Format of the data to be imported	CSV
is_security_cluster	Whether the security mode is used • **true** indicates that the security mode. • **false** indicates the normal mode.	true

Step 4 Save the modified **clickhouse_insert_tool.sh** file and run the following commands:

cd /opt/client

source bigdata_env

In security mode (Kerberos authentication is enabled), run the **kinit** command. In normal mode (Kerberos authentication is disabled), you do not need to run the following command:

kinit *clickhouseuser*

Step 5 Run the script to import data.

./ClickHouse/clickhouse_insert_tool/clickhouse_insert_tool.sh

Step 6 Log in to the ClickHouse client node and connect the server. For details, see **ClickHouse Client Practices**.

Step 7 Run the following command to query the distributed table corresponding to the local table where data is inserted and check the result:

select count(1) from testdb1.testtb1_all;

----End

3.4.9 Using ClickHouse to Import and Export Data

Using the ClickHouse Client to Import and Export Data

Use the ClickHouse client to import and export data.

- Importing data in CSV format

  clickhouse client --host *Host name or IP address of the ClickHouse instance* **--database** *Database name* **--port** *Port number* **--secure --format_csv_delimiter="***CSV file delimiter***" --query="INSERT INTO** *Table name* **FORMAT CSV" <** *Host path where the CSV file is stored*

  Example

  ```
  clickhouse client --host 10.5.208.5 --database testdb --port 9440 --secure --format_csv_delimiter="," --query="INSERT INTO testdb.csv_table FORMAT CSV" < /opt/data
  ```

  You need to create a table in advance.

- Exporting data in CSV format

  CAUTION

  Exporting data files in CSV format may cause CSV injection. Exercise caution when performing this operation.

  clickhouse client --host *Host name or IP address of the ClickHouse instance* **--database** *Database name* **--port** *Port number* **-m --secure --query=**"SELECT * **FROM** *Table name*" > *CSV file export path*

  Example

  ```
  clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="SELECT * FROM test_table" > /opt/test
  ```

- Importing data in Parquet format

  cat *Parquet file* **| clickhouse client --host** *Host name or IP address of the ClickHouse instance* **--database** *Database name* **--port** *Port number* **-m --secure --query="INSERT INTO** *Table name* **FORMAT Parquet"**

  Example


```
cat /opt/student.parquet | clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="INSERT INTO parquet_tab001 FORMAT Parquet"
```

- Exporting data in Parquet format

  clickhouse client --host *Host name or IP address of the ClickHouse instance* **--database** *Database name* **--port** *Port number* **-m --secure --query="select** * **from** *Table name* **FORMAT Parquet"** > *Parquet file export path*

  Example

  ```
  clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="select * from test_table FORMAT Parquet" > /opt/student.parquet
  ```

- Importing data in ORC format

  cat *ORC file path* **| clickhouse client --host** *Host name or IP address of the ClickHouse instance* **--database** *Database name* **--port** *Port number* **-m --secure --query="INSERT INTO** *Table name* **FORMAT ORC"**

  Example

  ```
  cat /opt/student.orc | clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="INSERT INTO orc_tab001 FORMAT ORC"
  # Data in the ORC file can be exported from HDFS. For example:
  hdfs dfs -cat /user/hive/warehouse/hivedb.db/emp_orc/000000_0_copy_1 | clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="INSERT INTO orc_tab001 FORMAT ORC"
  ```

- Exporting data in ORC format

  clickhouse client --host *Host name or IP address of the ClickHouse instance* **--database** *Database name* **--port** *Port number* **-m --secure --query="select** * **from** *Table name* **FORMAT ORC"** > *ORC file export path*

  Example

  ```
  clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="select * from csv_tab001 FORMAT ORC" > /opt/student.orc
  ```

- Importing data in JSON format

  INSERT INTO *Table name* **FORMAT JSONEachRow** *JSON string 1 JSON string 2*

  Example

  ```
  INSERT INTO test_table001 FORMAT JSONEachRow {"PageViews":5, "UserID":"4324182021466249494", "Duration":146,"Sign":-1}
  {"UserID":"4324182021466249494","PageViews":6,"Duration":185,"Sign":1}
  ```

- Exporting data in JSON format

  clickhouse client --host *Host name or IP address of the ClickHouse instance* **--database** *Database name* **--port** *Port number* **-m --secure --query="SELECT** * **FROM** *Table name* **FORMAT JSON|JSONEachRow|JSONCompact|..."** > *JSON file export path*

  Example

  ```
  # Export JSON file.
  clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="SELECT * FROM test_table FORMAT JSON" > /opt/test.json

  # Export json(JSONEachRow).
  clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="SELECT * FROM test_table FORMAT JSONEachRow" > /opt/test_jsoneachrow.json

  # Export json(JSONCompact).
  clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="SELECT * FROM test_table FORMAT JSONCompact" > /opt/test_jsoncompact.json
  ```

3.5 Enterprise-Class Enhancements of ClickHouse

3.5.1 ClickHouse Multi-Tenancy

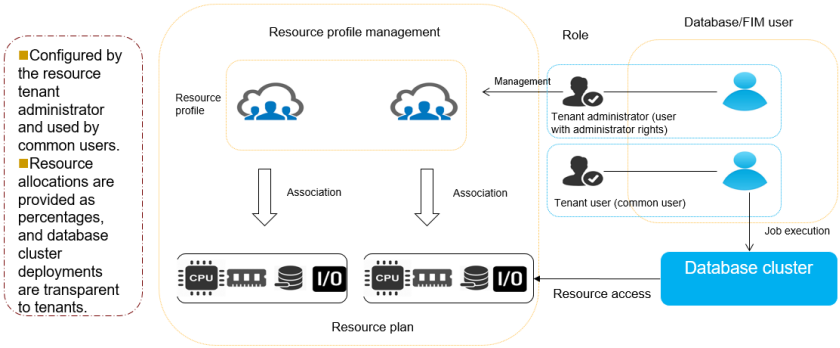
3.5.1.1 Overview

NOTE

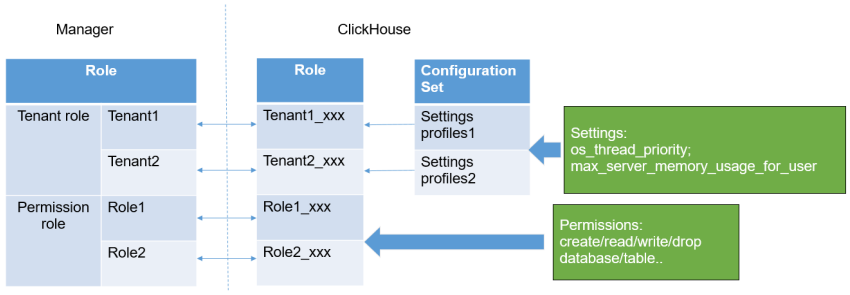
This section applies only to MRS 3.2.0 or later.

ClickHouse Multi-Tenancy

The ClickHouse multi-tenancy feature enables you to manage cluster resources through the user > tenant role > resource profile management model. Currently, memory and CPU priority management is supported. The following figure shows a multi-tenancy model.



On the service configuration and tenant management pages of FusionInsight Manager, you can configure memory quotas for services, create tenants, associate ClickHouse services, bind logical clusters, set available memory and CPU priorities for tenants, and associate tenants with users. The following figure illustrates the role association between Manager and ClickHouse.



The following table lists the resource configurations supported by the current version.

Resource	Value Range	Description	Remarks
Service-level memory resource limit	0–1	Percentage of available ClickHouse memory to total server memory	For example, if the physical memory of the server is 10 GB and the limit is set to **0.9**, the available memory of the ClickHouse service on the current server is 9 GB (10 GB x 0.9).
Tenant-level memory resource limit	0%–100%	Percentage of the available memory of the current tenant in ClickHouseServer	If this limit is set to **80**, the total memory that can be used by the current tenant is calculated as follows: Total memory that can be used by the service x 80%
Tenant-level CPU priority	–20 to 19	NICE value of the OS associated with this value. A smaller value indicates a higher CPU priority of the process.	This feature depends on **CAP_SYS_NICE** of the OS. By default, this feature is disabled after the cluster is installed. To use this feature, enable it by referring to **Configuring CPU Priority for ClickHouse Tenants**.

3.5.1.2 Configuring CPU Priority for ClickHouse Tenants

NOTE

This section applies only to MRS 3.2.0 or later.

Scenario

ClickHouse tenants support CPU priorities. This feature depends on CAP_SYS_NICE of the OS and takes effect only after being enabled.

Procedure

Step 1 Log in to the ClickHouseServer node as user **root** and run the following command:

setcap cap_sys_nice=+ep /opt/Bigdata/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/bin/clickhouse

NOTE

You need to run this command on all ClickHouseServer nodes.

Step 2 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click the **Instance** tab. Select all ClickHouseServer instances, and choose **More** > **Restart Instance**.

Step 3 Run the following command to check whether the CPU priority feature is enabled:

getcap /opt/Bigdata/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/bin/clickhouse

The following command output indicates that the feature has been enabled:

```
/opt/Bigdata/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse*/clickhouse/bin/clickhouse = cap_sys_nice+ep
```

----End

3.5.1.3 Creating a ClickHouse Tenant

NOTE

This section applies only to MRS 3.2.0 or later.

Scenario

On FusionInsight Manager, cluster administrators can create a ClickHouse tenant and associate it with a logical cluster. After a system user is bound to the tenant, the system user has the permissions on the logical cluster of the tenant.

Creating a ClickHouse Tenant

Step 1 Log in to FusionInsight Manager and choose **Tenant Resources**.

Step 2 Click ⊕. On the page that is displayed, configure tenant attributes according to **Table 3-7**.

Table 3-7 Tenant parameters

Parameter	Description
Cluster	Cluster for which you want to create a tenant
Name	• Name of the current tenant. The value consists of 3 to 50 characters, including numbers, letters, and underscores (_). • Plan a tenant name based on service requirements. The name cannot be the same as that of a role, HDFS directory, or Yarn queue that exists in the current cluster.
Tenant Resource Type	Select **Leaf Tenant Resource**. **NOTE** Create a ClickHouse tenant. **Tenant Resource Type** can only be **Leaf Tenant**.

Parameter	Description
Computing Resource	Dynamic compute resources for the current tenant • When **Yarn** is selected, the system automatically creates a queue in Yarn and the queue is named the same as the tenant name. • If **Yarn** is not selected, the system does not automatically create a queue.
Configuration Mode	Configuration mode of compute resource parameters • If you select **Basic**, you only need to configure **Default Resource Pool Capacity (%)**. • If you select **Advanced**, you can manually configure the resource allocation weight and the minimum, maximum, and reserved resources of the tenant.
Default Resource Pool Capacity (%)	Percentage of compute resources used by the current tenant in the default resource pool. The value ranges from 0 to 100%.
Weight	Resource allocation weight. The value ranges from 0 to 100.
Minimum Resource	Resources guaranteed for the tenant (preemption supported). The value can be a percentage or an absolute value of the parent tenant's resources. When a tenant has a light workload, the resources of the tenant are automatically allocated to other tenants. When the available tenant resources are less than the value of **Minimum Resource**, the tenant can preempt the resources that have been lent to other tenants.
Maximum Resource	Maximum resources that can be used by the tenant. The tenant cannot obtain more resources than the value configured. The value can be a percentage or an absolute value of the parent tenant's resources.
Reserved Resource	Resources reserved for the tenant resource. The reserved resources cannot be used by other tenants even if no job is running in the current tenant resources. The value can be a percentage or an absolute value of the parent tenant's resources.
Storage Resource	Storage resources for the current tenant. • When **HDFS** is selected, the system automatically allocates storage resources. • If **HDFS** is not selected, the system does not automatically allocate storage resources.
Quota	Quota for files and directories

Parameter	Description
Space Quota	Quota for the HDFS storage space used by the current tenant • If the unit is set to **MB**, the value ranges from 1 to 8796093022208. If the unit is set to **GB**, the value ranges from 1 to 8589934592. • This parameter indicates the maximum HDFS storage space that can be used by the tenant, but not the actual space used. • If its value is greater than the size of the HDFS physical disk, the maximum space available is the full space of the HDFS physical disk.
Storage Path	HDFS storage directory for the tenant • The system automatically creates a folder named after the tenant name in the **/tenant** directory by default. For example, the default HDFS storage directory for tenant **ta1** is **/tenant/ta1**. • When a tenant is created for the first time, the system automatically creates the **/tenant** directory in the HDFS root directory. The storage path is customizable.
Service	• Select **ClickHouse** for **Service**. – **Association Type**: When **Service** is set to **ClickHouse**, **Association Type** can only be set to **Shared**. – **Associate Logical Cluster**: If the logical cluster function is not enabled for ClickHouse, **default_cluster** is selected by default. If the function is enabled, select the logical cluster to which you want to associate. – **CPU Priority**: The CPU priority ranges from –20 to 19. This value is associated with the NICE value of the OS. A smaller value indicates a higher CPU priority. For details about how to enable the CPU priority, see **Configuring CPU Priority for ClickHouse Tenants**. – **Memory**: The maximum value of this parameter is **100**, in percentage. For example, if this parameter is set to **80**, the total memory that can be used by the current tenant is calculated as follows: Available memory x 80%.
Description	Description of the current tenant

Step 3 Click **OK**. Wait until the tenant is created.

Step 4 After the ClickHouse tenant is created, you can view and modify tenant resources on the **Tenant Resources** page.

1. On FusionInsight Manager, choose **Tenant Resources**. In the tenant list, select the ClickHouse tenant whose information you want to view and view the tenant overview and resource quota.
2. Choose **Resources** and click  next to **Resource Details** to modify tenant resources.
3. After the modification is complete, click **OK**. The modified resource details are displayed on the **Resources** page.

   NOTE

   After modifying the resource quota of the ClickHouse tenant, you need to log in to the ClickHouse client again for the modification to take effect.

----End

Adding a User and Binding the User to a Tenant

- To create a user and bind the user to a tenant, log in to FusionInsight Manager, choose **System** > **Permission** > **User**, click **Create User** to add a human-machine user, and add the tenant created by referring to **Creating a ClickHouse Tenant**. Then, the user has the permissions on the ClickHouse logical cluster.
- To bind an existing user to a tenant, log in to FusionInsight Manager, choose **System** > **Permission** > **User**, click **Modify** in the **Operation** column of the user, and add the tenant created by referring to **Creating a ClickHouse Tenant**. Delete the ClickHouse tenant from the role if the tenant is no longer needed.

  NOTE

  - After a user is bound to a ClickHouse tenant, the user has the permission to modify the logical cluster of the tenant.
  - When multiple users are bound to the same tenant, the tenant-level memory limit of the current version does not support real-time total memory limit. For example, user1 and user2 are bound to tenant1, the memory limit for tenant1 is 10 GB, and the query performed by user1 uses 5 GB memory. When user2 initiates a query, the memory that can be used by user2 is limited to 5 GB. During the query, the service does not dynamically update this restriction.
  - In the current version, a user cannot be bound to multiple ClickHouse tenants. If user1 has been associated with tenant1, no error message is displayed when user1 is associated with tenant2, but the information is recorded in background logs, indicating that the user has been associated with a tenant and this association operation is invalid.

Associating an Existing Tenant with the ClickHouse Service

1. On FusionInsight Manager, choose **Tenant Resources**, select the tenant to which you want to associate a service, click the **Service Association** tab, and click **Associate Service**. The following table describes the parameters.

Parameter	Description
Service	Select **ClickHouse**.
Association Type	Select **Shared**.
Associate Logical Cluster	If the logical cluster function is not enabled for ClickHouse, **default_cluster** is selected by default. If the function is enabled, select the logical cluster to which you want to associate.
CPU Priority	The CPU priority ranges from –20 to 19. This value is associated with the NICE value of the OS. A smaller value indicates a higher CPU priority. For details about how to enable the CPU priority, see **Configuring CPU Priority for ClickHouse Tenants**.
Memory	The maximum value of this parameter is **100**, in percentage. For example, if this parameter is set to **80**, the total memory that can be used by the current tenant is calculated as follows: Available memory x 80%.

2. On the displayed tab page, configure the tenant based on service requirements and click **OK**. The tenant is associated with the service.
3. To disassociate the ClickHouse service, perform the following operations:

   On FusionInsight Manager, choose **Tenant Resources**, select the tenant whose ClickHouse service is to be disassociated, and click **Delete** in the **Operation** column. In the dialog box that is displayed, click **OK**.

   NOTE

   After the ClickHouse service is disassociated from a tenant, the tenant and its users no longer have the permissions on the ClickHouse logical cluster.

3.5.1.4 Modifying the Memory Limit of ClickHouse on a ClickHouseServer Node

NOTE

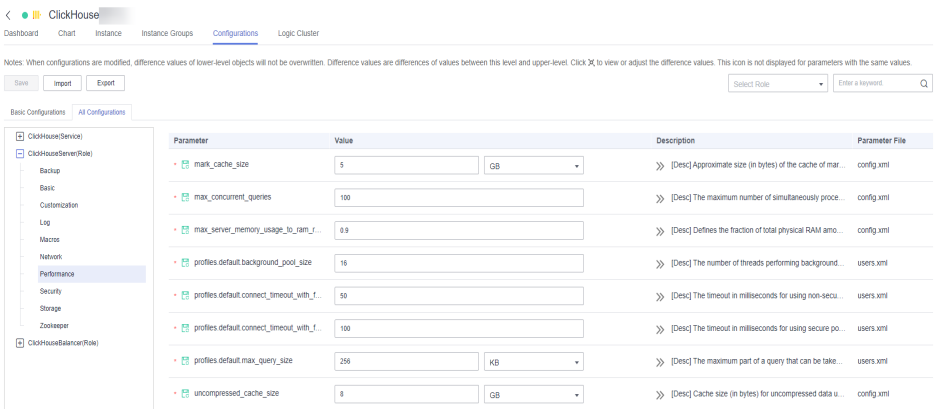
This section applies only to MRS 3.2.0 or later.

Scenario

Modify the maximum memory allowed for ClickHouse on a ClickHouseServer node to ensure the normal use of other service instances on the node.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Configuration** then **All Configurations**, click **ClickHouseServer(Role)**, and select **Performance**.



Step 2 Change the value of **max_server_memory_usage_to_ram_ratio** as required and save the configuration.

NOTE

- Restart is not required for the modification to take effect.
- The value ranges from 0 to 1, indicating the ratio of the total physical RAM of the server that can be used for ClickHouse. For example, if the physical memory of the server is 10 GB and the value of this parameter is **0.9**, the available memory of the ClickHouse service on the current server is 9 GB (10 GB x 0.9). If the value of this parameter is **0**, it indicates that the memory is not limited and the ClickHouse service can use all the physical memory of the server. The value of this parameter can contain a maximum of two decimal places.

----End

3.5.2 Checking Slow SQL Statements in ClickHouse

Scenario

The SQL statement query in ClickHouse is slow because the conditions such as partitions, where conditions, and indexes of SQL statements are set improperly. As a result, the overall performance of the database is affected. To solve this problem, MRS provides the function of monitoring slow ClickHouse query statements.

Ongoing Slow Queries

You can query information about slow SQL statements that are being executed but do not return any result.

- **Procedure**

  For versions earlier than MRS 3.2.0, log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. On the displayed page, click the **Query Management** tab and then the **Ongoing Slow Queries** tab.

  Log in to FusionInsight Manager of an MRS 3.2.0 cluster or later, click **Cluster**, choose **Services** > **ClickHouse**, click **Logic Cluster**, and click the name of the target logical cluster. On the displayed page, choose **Query Management** > **Ongoing Slow Queries**.

- **Parameters**

Table 3-8 Slow query parameters

Parameter	Description
Server Node IP Address	IP address of the ClickHouseServer instance. To view the IP address, log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. On the displayed page, click the **Instance** tab.
Query ID	Unique ID generated internally.
Query	Slow query SQL statement.
Start Time	Time when the execution of a slow query SQL statement starts.
End Time	Time when the execution of a slow query SQL statement ends.
Duration (s)	Total execution time of a slow query SQL statement, in seconds.
User	ClickHouse user who executes a slow query SQL statement.
Client IP Address	IP address of the client that submits a slow query SQL statement.
Memory Used (MB)	Memory used by a slow query SQL statement, in MB.
Operation	You can click **Terminate** to terminate the slow query using a slow query SQL statement.

- **Filter conditions**

  Select the query condition as required and filter the query results.

  Table 3-9 Filter conditions

Condition	Description
Slow query duration exceeding	Filters the slow queries based on the duration. The value can be **3 (s)**, **9 (s)**, **15 (s)**, or **25 (s)**.
By Query ID	Filters the slow queries based on the query ID. Fuzzy search based on the query ID is supported. For example, if the query ID is **111-222-333-444-555**, you can enter part of it such as **111-222** or **-222-333** to query the needed information.
By User	Filters the slow queries based on the ClickHouse user. Fuzzy search based on part of a username is supported.

Condition	Description
By Client IP Address	Filters the slow queries based on the IP address of the client that submits slow query SQL statements. Fuzzy search based on part of the client IP address is supported. For example, if the client IP address is **192.168.0.1**, you can enter part of it such as **192.168** or **192.168.0** to query the information you need.

Completed Queries

You can query information about slow SQL statements that have been executed and returned results.

UI path:

Log in to FusionInsight Manager of a cluster earlier than MRS 3.2.0 and choose **Cluster** > **Services** > **ClickHouse**. On the displayed page, click the **Query Management** tab and then the **Completed Queries** tab.

Log in to FusionInsight Manager of an MRS 3.2.0 cluster or later, click **Cluster**, choose **Services** > **ClickHouse**, click **Logic Cluster**, and click the name of the target logical cluster. On the displayed page, choose **Query Management** > **Completed Queries**.

For details about slow query parameters and filter conditions, see **Table 3-8** and **Table 3-9**, respectively.

3.5.3 Checking Monitoring Metrics of ClickHouse Replication Table Data Synchronization

Scenario

MRS monitors the synchronization between multiple replicas of data in the same shard of a Replicated*MergeTree table.

Constraints

Currently, you can monitor and query only Replicated*MergeTree tables whose creation statements contain the key word **ON CLUSTER**.

Replication Table Data Synchronization

- **Procedure**

  Log in to FusionInsight Manager of a cluster earlier than MRS 3.2.0, click **Cluster**, choose **Services** > **ClickHouse**, and click **Data Synchronization Status**.

  Log in to FusionInsight Manager of an MRS 3.2.0 cluster or later, click **Cluster**, choose **Services** > **ClickHouse**, click **Logic Cluster**, and click the name of the

target logical cluster. Go to the logical cluster page and select **Data Synchronization Status**.

- **Data synchronization parameters**

  Table 3-10 Data synchronization parameters

Parameter	Description
Data Tables	Names of Replicated*MergeTree tables.
Database	Database where the data table is located.
Shard	ClickHouse shard where the data table is located.
Status	Data synchronization status. The options are as follows: • **No data**: The table has no data on this shard. • **Synchronized**: The table has data on this shard, and multiple instance copies of the same shard are the same. • **Not synchronized**: The table has data on this shard, but multiple instance copies of the same shard are not the same.
Details	Data synchronization details of the data table on the corresponding ClickHouseServer instance.

- **Filter conditions**

  You can select **By Data Tables** and enter a data table name in the search box to filter data tables.

3.5.4 Configuring Strong Data Consistency Between ClickHouse Replicas

NOTE

This topic is available for MRS 3.3.0-LTS and later versions only.

Scenario

ClickHouse supports multiple replicas. When data is written to a local table, the data on the current node is updated immediately, but the data between other replicas is asynchronously updated.

Configure ClickHouse to ensure strong data consistency between replicas.

Parameters

NOTE

The priority of configuring strong data consistency between ClickHouse replicas is as follows: single statements > sessions > global defaults.

Strong data consistency between replicas must be used together with atomicity. Otherwise, an exception occurs during data insertion and the rollback fails.

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** then **All Configurations**, click **Reliability**, and search for and modify the following parameters.

Parameter	Description
profiles.default.insert_quorum	Whether to enable strong data consistency between ClickHouse replicas. The default value is **0**, indicating that strong consistency between replicas is disabled. Value options are as follows: 0 to 9, **auto**, and **all**. • If this parameter is set to a number, the INSERT operation can succeed only when ClickHouse attempts to correctly write data to the **insert_quorum** of replicas during the **insert_quorum_timeout** period. • **auto** indicates that the INSERT operation is successful only when data is correctly written to more than half of the replicas. • **all** indicates that the INSERT operation is successful only when data is correctly written to all replicas.
profiles.default.insert_quorum_timeout	Timeout interval for writing strongly consistent data between replicas. The default value is **600000** ms. The value must be greater than 0.

3.5.5 Configuring the Support for Transactions on ClickHouse

NOTE

This topic is available for MRS 3.3.0-LTS and later versions only.

Scenario

Atomicity means that a transaction is an inseparable unit of work. A transaction can contain multiple operations, which are either all executed or none executed. However, some exceptions may occur during transaction execution, for example, a user rolls back a transaction, a connection is disconnected, or a power failure occurs. As a result, the transaction execution is interrupted.

ClickHouse supports atomic write and transaction capabilities. The atomicity of a transaction means that after an operation of a transaction fails, the transaction can be rolled back to the state before the transaction is executed.

Start a ClickHouse transaction.

NOTE

- Writing data to local tables achieves better performance. So, multi-replica distributed transactions are recommended for adding, deleting, modifying, and querying data on local tables.
- When data is written to a distributed table, the distributed table transaction **insert_distributed_sync** and local table transaction **Mergetree/ReplicateMergeTree** must be combined to support data writing.

Parameters

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** then **All Configurations**, click **Reliability**, and search for and modify the following parameters.

Parameter	Description
allow_transactions	Whether to support transactions. The value can be **0** or **1**. • The default value is **0**, indicating that transactions are not supported. • Set this parameter to **1**, save the configuration, and restart the service for the support for transactions to take effect.
_clickhouse.metrika.cluster.internal_replication	Whether to write data to only one replica. The value can be **true** or **false**. • The default value is **true**, indicating that data is inserted only to one replica. • If this parameter is set to **false**, data is inserted to both replicas.

NOTE

- You can run the **set implicit_transaction='true';** statement to use session-level implicit transactions. Currently, ClickHouse does not support interruption of alter queries. If the execution of alter queries (for example, lightweight delete) is interrupted, it cannot be rolled back even if implicit transactions are enabled. This is the same as open-source ClickHouse.
- To insert data into a distributed table, perform the following steps:

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse**, click **Configurations** then **All Configurations**, and change the value of **_clickhouse.metrika.cluster.internal_replication** to **false**. This means that data is written to all replicas of a shard when being inserted to a distributed table.

  At the session level, **set insert_distributed_sync='true';** indicates that data is inserted to each actual table in synchronous mode when being inserted to a distributed table.

3.5.6 Accessing ClickHouse Through ELB

Currently, ClickHouse is deployed in cluster mode regardless of whether replication or sharding is used. When ClickHouse provides services externally, multiple ClickHouse service nodes will be exposed and no unified access entry is available. ClickHouse provides the BalancedClickhouseDataSource class, which supports the load balancing capability by randomly allocating client's access requests to multiple nodes. However, it behaves unsatisfactorily in fault detection. Especially, the client needs to proactively detect changes of cluster nodes during scale-in or scale-out.

MRS work with Elastic Load Balance (ELB) to address the preceding issues. **Figure 3-4** shows the deployment architecture. This architecture can automatically distribute user access traffic evenly to multiple backend ClickHouse nodes, expanding external service capabilities and improving fault tolerance. When a backend ClickHouse node becomes faulty, ELB automatically fails over access traffic to another properly running node.

Figure 3-4 Accessing ClickHouse nodes through ELB

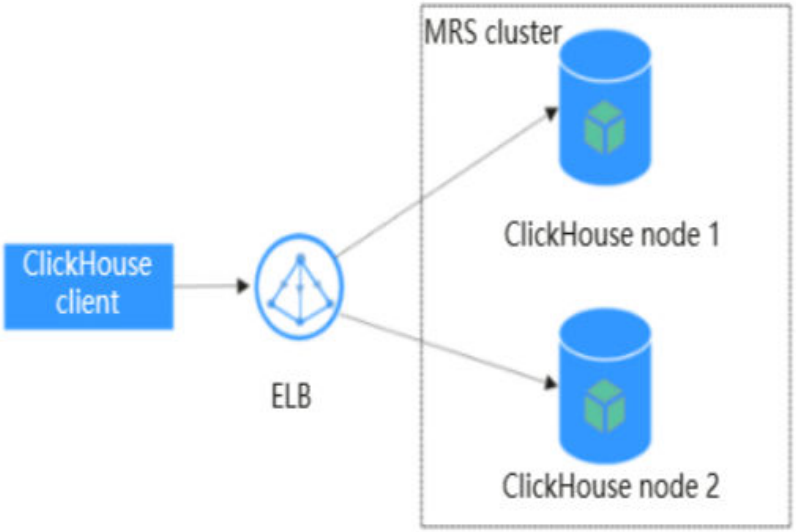


Table 3-11 lists the advantages of the ELB-based deployment over BalancedClickhouseDataSource.

Table 3-11 Differences between ELB-based deployment and BalancedClickhouse-DataSource

Load Balancing Method	Difference
ELB-based deployment	• Supports multiple request policies. • Supports automatic fault detection and failover. • Allows you to add ClickHouse backend nodes simply by changing ELB configurations.

Load Balancing Method	Difference
BalancedClickhouseData-Source	• Causes load imbalance due to random allocation of requests. • Lacks of sufficient fault detection capabilities.

Table 3-12 lists the supported protocols and ports for accessing ClickHouse through ELB. Configure them as required.

Table 3-12 Supported protocols and ports for accessing ClickHouse nodes through ELB

Protocol	Port Number	Used When
TCP	9000	A client request is sent to ELB to connect to ClickHouse. For example, if you run the **clickhouse client** command to connect to ClickHouse, set **host** to the private IP address of ELB.
HTTP	8123	An HTTP request is sent to ELB to connect to ClickHouse.

This section describes how to use a client to access ClickHouse through ELB. The procedure is as follows:

- **Step 1: Buy an ELB and obtain its private IP address.**
- **Step 2: Add an ELB listener and configure its protocol and port.**
- **Step 3: Add backend ClickHouse servers to the ELB.**
- **Step 4: Use a client to access ClickHouse through ELB.**

Prerequisites

- You have created an MRS cluster and its ClickHouse instances are running properly.
- You have installed the MRS client, for example, in the **/opt/client** directory. The client directory in the following operations is only an example. Change it to the actual installation directory.

Buying an ELB and Connecting It to ClickHouse Nodes

Buying an ELB and obtaining its private IP address

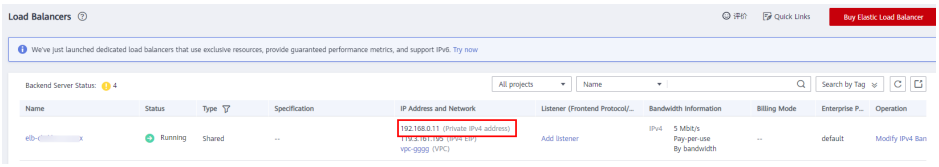
For details, see **Creating a Shared Load Balancer**.

Step 1 Log in to the ELB console and click **Buy Elastic Load Balancer**.

Step 2 On the **Buy Elastic Load Balancer** page, set **Type** to **Shared**, set **VPC** and **Subnet** to the same values as those of the MRS cluster, and retain the default values for other parameters.

Step 3 Click **Next**, confirm the configurations, and click **Submit**.

Step 4 On the **Load Balancers** page, obtain the private IP address of the newly created load balancer.



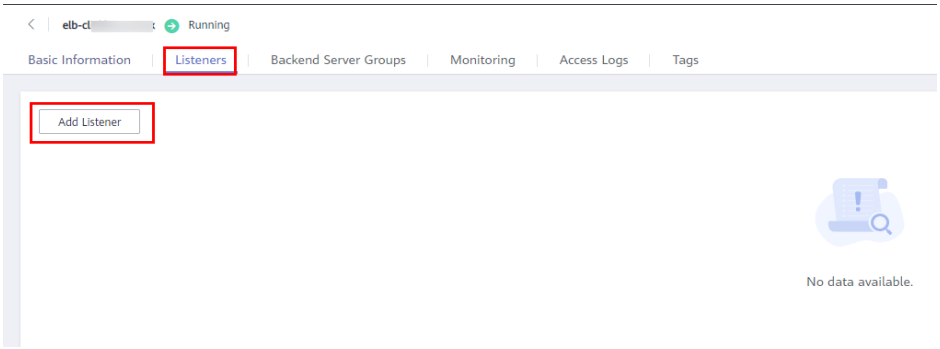
----End

Adding an ELB listener

For details, see **Adding a TCP Listener**.

Step 1 On the **Load Balancers** page, click the name of the created load balancer to go to its details page.

Step 2 Click the **Listeners** tab and then **Add Listener**.



Step 3 On the **Add Listener** page, complete the configuration as prompted.

1. Configure the listener.

   Set **Frontend Protocol/Port** to **TCP** and **9000**, respectively. Retain the default values for other parameters. Click **Next**.

   NOTE

   If an HTTP request is sent to access ClickHouse through ELB, set **Frontend Protocol/Port** to **HTTP** and **8123**, respectively.

2. Configure the backend server group.

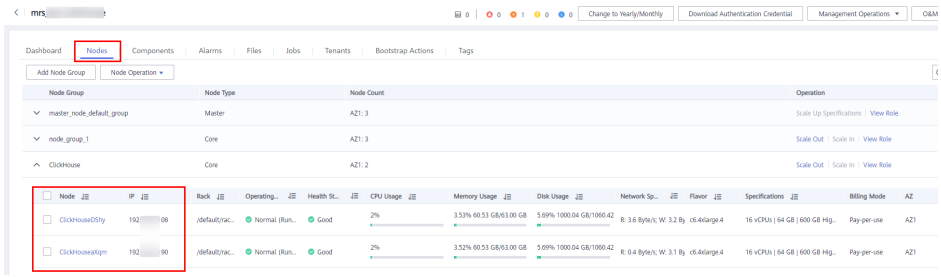
   Set **Load Balancing Algorithm** to **Weighted round robin** and click **Finish**. On the displayed page, click **OK**.

----End

Adding ClickHouse backend servers

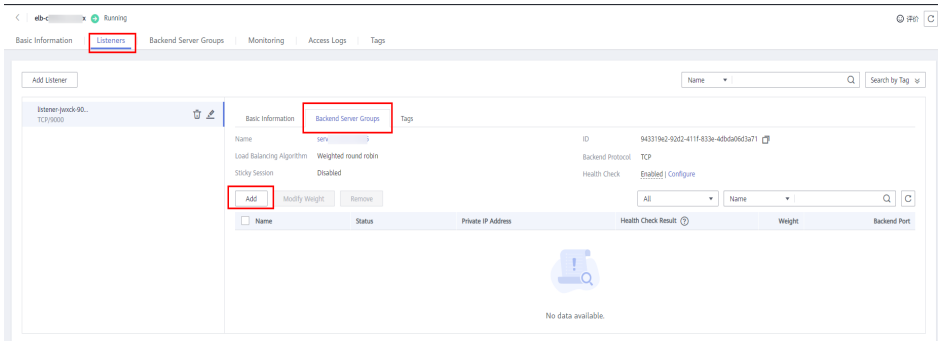
Step 1 Switch to the MRS console and click the MRS cluster to be interconnected.

Step 2 On the displayed page, click the **Nodes** tab and expand **ClickHouse** to obtain its node names and IP addresses.



Step 3 Switch to the ELB console, locate the created load balancer, and click its name.

Step 4 Click the **Listeners** tab and then **Backend Server Groups**. Click **Add**.



Step 5 On the **Add Backend Server** page, select the backend servers based on the node names and IP addresses of ClickHouse obtained in **Step 2**. Click **Next**.

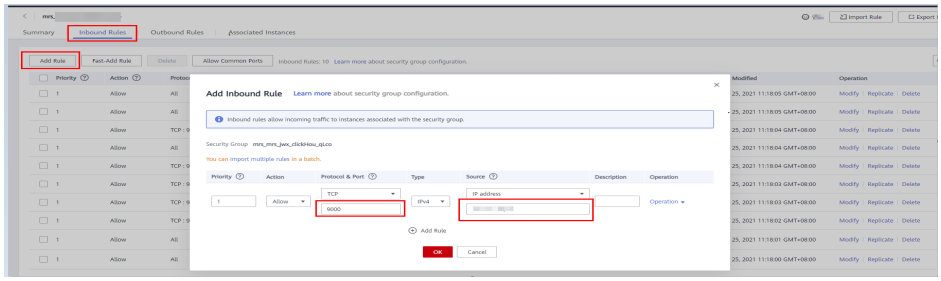
Step 6 On the displayed page, set **Batch Add Ports** to **9000** and click **OK**. Confirm your configurations and click **Finish**.

NOTE

If an HTTP request is sent to access ClickHouse through ELB, set **Batch Add Ports** to **8123**.

Step 7 Configure the security group.

After the configuration is complete, go to the **Backend Server Groups** tab on the **Listeners** page. The **Health Check Result** of the backend servers is **Unhealthy**.



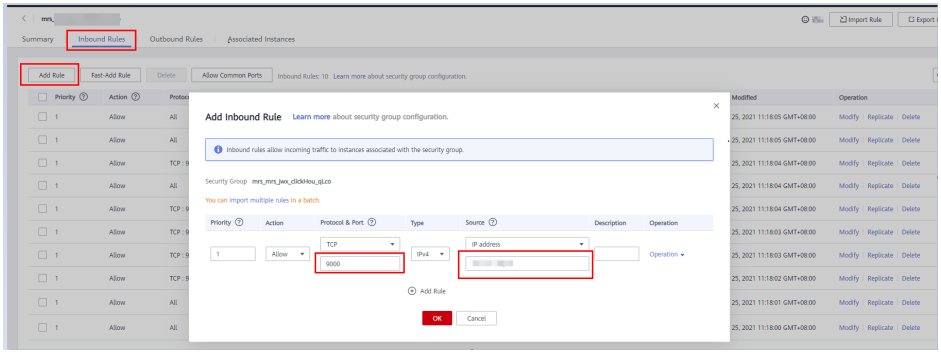
To solve this issue, you need to configure the inbound rule of the security group for the ClickHouse backend server to allow access from 100.125.0.0/16. The procedure is as follows:

1. Go back to the **Listeners** tab. Click the **Backend Server Groups** subtab and then the name of a backend server.
2. On the displayed page, click the **Security Groups** tab and then **Manage Rule**. Next, click the **Inbound Rules** tab and then **Add Rule**.

3. On the **Add Inbound Rule** page, set **Protocol & Port** to **TCP** and **9000**, and **IP address** to **100.125.0.0/16**. Click **OK**.

   NOTE

   If an HTTP request is sent to access ClickHouse through ELB, set **Batch Add Ports** to **8123**.



4. Go back to the ELB console, navigate to the details page of the created load balancer, and refresh the page. Click the **Listeners** tab and then the **Backend Server Groups** subtab. The **Health Check Status** changes to **Healthy**.

----End

Accessing ClickHouse through ELB

Step 1 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse** > **Configurations** > **All Configurations**, and set **SSL_NONESSL_BOTH_ENABLE** to **true**.

Step 2 Use the client to log in to the node where the ClickHouse service instance is deployed. For details, see **ClickHouse Client Practices**. Note that the **host** parameter in the **clickhouse client** command must be set to the private IP address of ELB.

Step 3 Check the connection result on the client.

NOTE

If you manually run a client command to connect to the ClickHouse node, there may be only a few concurrent requests. In this case, the ELB may always send the requests to the same backend ClickHouse node. This is normal.

If there are a large number of concurrent requests, the ELB will distribute the requests to multiple ClickHouse nodes in polling mode.

----End

3.5.7 Storing ClickHouse Cold and Hot Data Separately

NOTE

This section applies to MRS 3.3.1-LTS or later.

Scenario

Based on the multi-volume storage of the open-source ClickHouse, MRS allows ClickHouse tables to be stored in volumes of multiple devices. You can define

different types of disks in a volume and store the cold and hot data separately. Cold data is stored in OBS, and hot data is stored in ClickHouse. ClickHouse clusters provide optimal query performance and long-term data storage at a low cost.

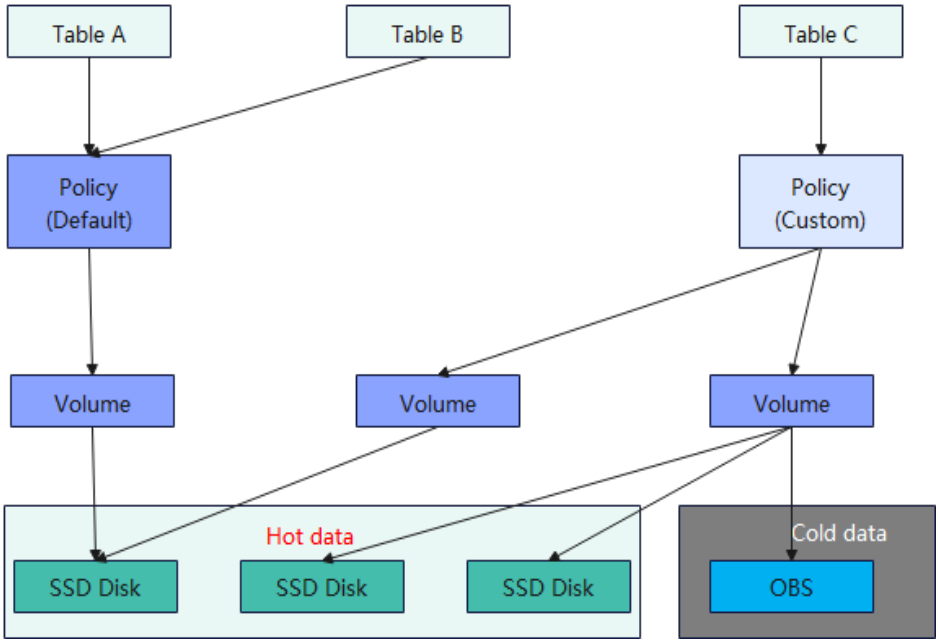
The process of configuring separated storage for cold and hot data in ClickHouse is as follows:

1. **Creating an OBS Parallel File System**
2. **Creating a Cloud Service Agency and Binding It to a Cluster**
3. **Creating a Common Account Agency and Binding It to a Cluster**
4. **Adding OBS Disk Information to the ClickHouse Cluster**
5. **Configuring Disk Storage Policies**
6. **Custom Cold-Hot Separation Policy**
7. **Importing Data into ClickHouse to Verify the Configuration**

Principles

OBS is a secure, reliable, and cost-effective distributed storage service that can handle large-scale data. ClickHouse adopts a cold and hot data separated architecture on top of OBS advantages. The SSD of the node where the ClickHouse instance is deployed stores the hot data (recently generated and frequently accessed). OBS stores the cold data (accessed less frequently than hot data). When creating a table, use TTL (a time policy) to store cold data and hot data separately.

Figure 3-5 Working principle of cold and hot data storage



Terms you need to know are described as follows:

Volume: a collection of ordered disks

Storage Policy: a storage policy, a collection of volumes, and rules for moving data among volumes

Creating an OBS Parallel File System

1. Log in to the management console.
2. Click **Service List** and choose **Storage** > **Object Storage Service**.
3. Click **Parallel File System** and click **Create Parallel File System**. On the displayed page, set the following parameters, and click **Create Now**.
   - **Region**: Select the region where the MRS cluster is located.
   - **File System Name**: Enter a file system name, for example, **mrs-ck-obs**.
4. Click the name of the created parallel file system. On the **Overview** page, view and record the value of **Endpoint**, which is the endpoint of OBS.
5. Obtain the IAM endpoint.
   - To obtain the system endpoint, remove the prefix of the OBS endpoint, that is, **obs.**.
   - The endpoint of IAM is **iam.***System endpoint*.

   For example, if the endpoint of OBS is **obs.xxx.huawei.com**, the system endpoint is **xxx.huawei.com** and the IAM endpoint is **iam.cxxx.huawei.com**.
6. Obtain the AK/SK information.
   a. Move the cursor to the login user name in the upper right corner and select **My Credentials** from the drop-down list.
   b. On the **API Credentials** page, obtain the **Account ID** whish is used as the domain ID.
   c. Click **Access Keys**, click **Create Access Key**, and enter the verification code or password. Click **OK** and download the access keys. Obtain the AK/SK information from the **.csv** file.

      NOTE

      After a service is deleted or a cluster is uninstalled, dirty data may remain in the parallel file system **3**. You need to delete the dirty data.

Creating a Cloud Service Agency and Binding It to a Cluster

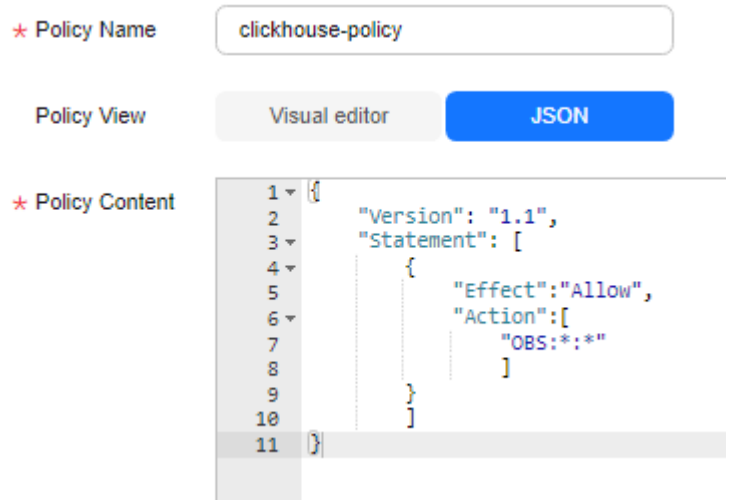
1. Log in to the management console.
2. In the service list, choose **Management & Governance** > **Identity and Access Management**.
3. In the navigation pane on the left, choose **Agencies** and click **Create Agency**. On the displayed page, set the following parameters and click **Next**:
   - **Agency Name**: Enter an agency name, for example, **mrs_clickhouse_obs**.
   - **Agency Type**: Select **Cloud service**.
   - **Cloud Service**: Select **ECS BMS**.
   - **Validity Period**: Select **Unlimited**.
4. On the displayed page, search for the **OBS OperateAccess** policy and select it.
5. Click **Next**, click **Show More**, select **Global services**, and click **OK**.
6. In the displayed dialog box, click **OK** to start authorization. Click **Finish** after the message "Authorization successful." is displayed.

7. On the MRS console, choose **Active Clusters** in the navigation pane on the left and click the name of the target cluster to access its details page.
8. On the **Dashboard** page, click **Synchronize** on the right of **IAM User Sync** to synchronize the IAM user.
9. Click **Manage Agency** on the right of **Agency**, select the created agency, for example, **mrs_clickhouse_obs**, and click **OK** to bind the agency to the cluster.

Creating a Common Account Agency and Binding It to a Cluster

1. Log in to the management console.
2. In the service list, choose **Management & Governance** > **Identity and Access Management**.
3. Choose **Permissions** > **Policies/Roles**, and click **Create Custom Policy**. Set the following parameters, and click **OK**:
   - **Policy Name**: Enter a policy name, for example, **clickhouse-policy**.
   - **Policy View**: Select **JSON**.
   - **Policy Content**: Enter the following content:

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Effect":"Allow",
         "Action":[
            "OBS:*:*"
           ]
      }
      ]
}
```



4. In the navigation pane on the left, click **Agencies**. Click **Create Agency**. On the displayed page, set the following parameters and click **Next**:
   - **Agency Name**: Enter an agency name, for example, **agency-clickhouse-to-OBS**.
   - **Agency Type**: Select **Account**.
   - Enter your cloud account in the **Delegated Account** field, that is, the account you register using your mobile phone number. It cannot be a federated user or an IAM user created using your cloud account.

- **Validity Period**: Select **Unlimited**.

5. In the search box on the displayed **Authorize Agency** page, search for the custom policy created in **3** and select it, for example, **clickhouse-policy**.
6. Click **Next**, click **Show More**, select **Global services**, and click **OK**.
7. Check and record the agency ID.
8. On the **Identity and Access Management** page, click **Agencies**.
9. Click the name of the cloud service agency created in **3**, for example, **mrs_clickhouse_obs**.
10. Click the **Permissions** tab and click **Authorize**. On the displayed page, click **Create Policy** in the upper right corner, and set the parameters as follows:
    - **Policy Name**: Enter a policy name, for example, **clickhouse-assume-policy**.
    - **Policy View**: Select **JSON**.
    - **Policy Content**: Enter the following content. *{Agency ID}* indicates the ID recorded in **7**.

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Action": [
            "iam:agencies:assume"
         ],
         "Resource": {
            "uri": [
               "/iam/agencies/{Agency ID}"
            ]
         },
         "Effect": "Allow"
      }
   ]
}
```

11. Click **Next**. On the **Select Policy/Role** page, select the policy created in **10**.
12. Click **Next**, click **Show More**, select **Global services**, and click **OK**.

Adding OBS Disk Information to the ClickHouse Cluster

1. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** then **All Configurations**, click **ClickHouseServer(Role)**, and select **Storage**.
2. OBS access credentials can be an agency or AK/SK. Choose either of them.
    - Configure an agency to obtain OBS access credentials.

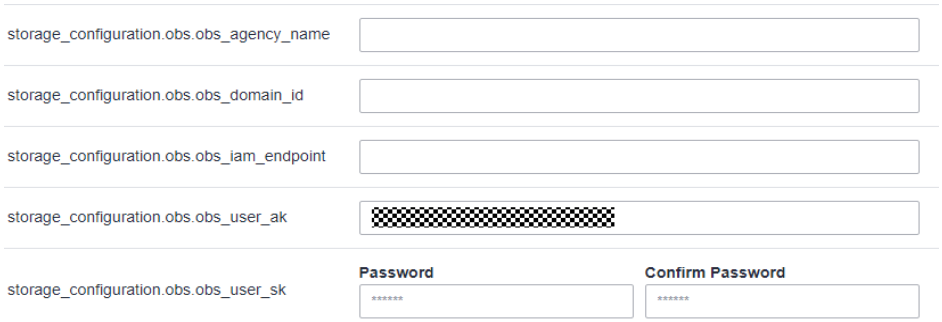
Parameter	Description
storage_configuration.obs.obs_domain_id	Domain ID of the user who accesses IAM For details, see the domain ID obtained in **Creating an OBS Parallel File System**.
storage_configuration.obs.obs_iam_endpoint	For details, see the IAM endpoint obtained in **Creating an OBS Parallel File System**.

Parameter	Description
storage_configuration.obs.obs_agency_name	Agency name. For details about how to obtain the agency name, see **Creating a Common Account Agency and Binding It to a Cluster**. Example value: **agency-clickhouse-to-OBS**

- Use AK/SK as the OBS access credentials.

Parameter	Description
storage_configuration.obs.obs_user_ak	AK of the IAM user in prerequisites To obtain the AK information, refer to **6**.
storage_configuration.obs.obs_user_sk	SK of the IAM user in prerequisites To obtain the SK information, refer to **6**.

Figure 3-6 Adding AK/SK to OBS credentials



3. In the **clickhouse-config-customize** parameter, add custom configuration items based on the following table to configure OBS disk information.

Table 3-13 OBS disk parameters

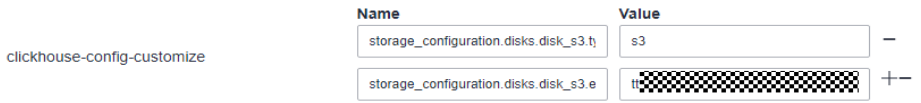
Parameter	Description
storage_configuration.disks.*disk_s3*.type	Fixed at **s3**

Parameter	Description
storage_configuration.disks.*disk_s3*.endpoint	Access path of the created OBS parallel file system. The format is as follows: *https://Parallel file system name.Parallel File Endpoint/Folder Name/* • Parallel file system name: Name of the OBS parallel file system you cerated • Parallel file endpoint: To obtain the endpoint, click the OBS parallel file name and check the basic information on the overview page. • Folder name: name of the folder created in the OBS parallel file system

NOTE

disk_s3 indicates the OBS disk name, which can be customized.

Figure 3-7 Configuring an OBS disk with **clickhouse-config-customize**



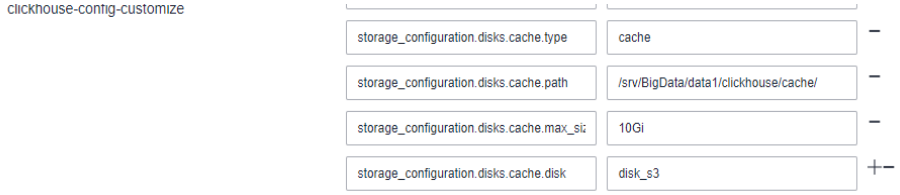
4. (Optional) Cache data stored on OBS to local storage to accelerate data query. To use the local cache, add configuration items to **clickhouse-config-customize**.

Parameter	Description
storage_configuration.disks.*obs_cache*.type	Fixed at **cache**
storage_configuration.disks.*obs_cache*.path	Path for caching data on a local disk, for example, **/srv/BigData/data1/clickhouse/cache/**
storage_configuration.disks.*obs_cache*.max_size	Cache size, for example, 10 Gi
storage_configuration.disks.*obs_cache*.disk	Name of the OBS storage disk, for example, **disk_s3**

NOTE

obs_cache indicates the OBS cache name, which can be customized.

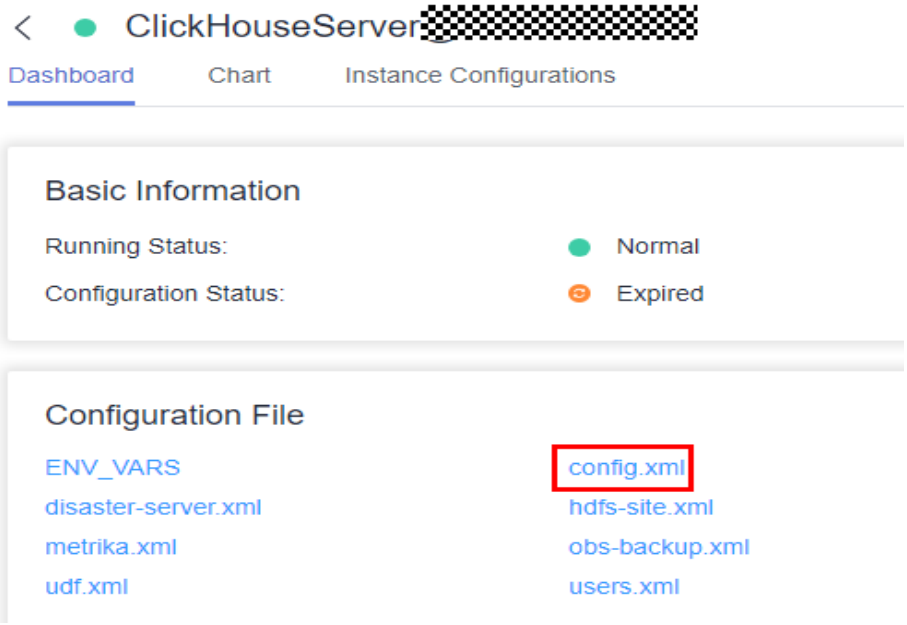
Figure 3-8 Configuring an OBS cache with **clickhouse-config-customize**



Configuring Disk Storage Policies

Configure disk storage policies on FusionInsight Manager.

1. Log in to FusionInsight Manager, choose **Cluster** > **ClickHouse** > **Instances**, and click a **ClickHouseServer**. On the displayed page, click the **config.xml** file.



2. In the **config.xml** file, obtain the value of **disk** of **storage_configuration**.


```
<grpc_port/>
<include_from>/opt/huawei/Bigdata/FusionInsight_ClickHouse_8.1.3/1_2_ClickHouseServer/e
<remote_servers incl="clickhouse_remote_servers"/>
<dictionaries_config>*_dictionary.xml</dictionaries_config>
<tcp_with_proxy_port/>
<users_config>users.xml</users_config>
<listen_host>server-2110081635-0001</listen_host>
<default_profile>default</default_profile>
<mark_cache_size>5368709120</mark_cache_size>
<storage_configuration>
  <disks>
    <disk1>
      <path>/srv/BigData/clickhouse/data1/clickhouse/</path>
      <keep_free_space_bytes>104857600</keep_free_space_bytes>
    </disk1>
  </disks>
  <policies>
    <default>
      <volumes>
        <volume1>
          <disk>disk1</disk>
          <max_data_part_size_bytes>0</max_data_part_size_bytes>
        </volume1>
      </volumes>
    </default>
  </policies>
</storage_configuration>
<zookeeper>
  <node index="1">
    <host>server-2110082001-0019</host>
```

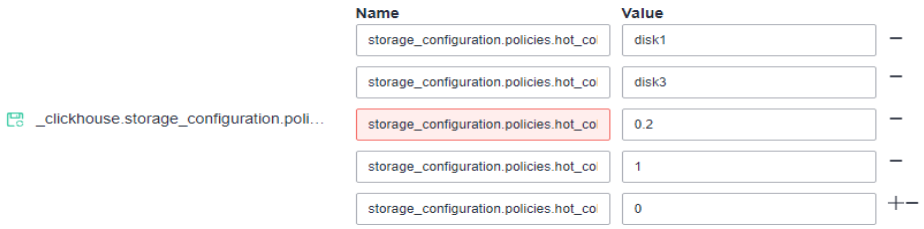
3. Choose **Cluster** > **ClickHouse** > **Configurations** > **All Configurations** > **ClickHouse(Service)** and search for the **_clickhouse.storage_configuration.policies** parameter.

   Set the following parameters listed in the table.

Configuration Item	Value	Description
storage_configuration.policies.*hot_cold_separation_policy*.volumes.*hot_volume*.disk[1]	disk1	This is a configuration item for local disks. You can configure multiple local disks as you need. The value ranges from **disk1** to **diskn**. For details about how to obtain the parameter value, see 2. **hot_cold_separation_policy** indicates the name of a policy item, which can be customized. **hot_volume** indicates the volume name in the hot data storage policy, which can be customized.
storage_configuration.policies.*hot_cold_separation_policy*.volumes.*hot_volume*.disk[2]	disk2	
storage_configuration.policies.*hot_cold_separation_policy*.volumes.*hot_volume*.disk[n]	diskn	

Configuration Item	Value	Description
storage_configuration.policies.*hot_cold_separation_policy*.volumes.hot_volume.priority	1	Priority of volumes in the hot data storage policy. If the value is greater than that of **storage_configuration.policies.*hot_cold_separation_policy*.volumes.cold_obs_volume.priority**, the hot data disk has a higher priority than the cold data disk.
storage_configuration.policies.*hot_cold_separation_policy*.volumes.cold_obs_volume.priority	0	Priority of volumes in the cold data storage policy. If the value is greater than that of **storage_configuration.policies.*hot_cold_separation_policy*.volumes.hot_volume.priority**, the cold data disk has a higher priority than the hot data disk.
storage_configuration.policies.*hot_cold_separation_policy*.volumes.*cold_obs_volume*.disk	OBS disk or cache name	**cold_obs_volume** indicates the volume name in the cold data OBS storage policy, which can be customized. ● If you do not use the local disk cache, set this parameter to the name of the OBS disk, for example, **disk_s3**. ● If you are using the local disk cache, set this parameter to the cache name, for example, **obs_cache**.
storage_configuration.policies.*hot_cold_separation_policy*.move_factor	0.2	**move_factor** indicates when data is moved to another volume. When the available space of a volume is less than **move_factor**, data is automatically moved to the next volume. In this example, when the available space of the hot volume is less than 20%, data in this volume is automatically moved to the cold volume. The value range is 0 to 1.

The following figure shows how to configure a storage policy.



4. Save the configuration and restart ClickHouse.

Custom Cold-Hot Separation Policy

1. Run the **clickhouse client** command to connect to the ClickHouseServer node. For details, see **ClickHouse Client Practices**.
2. Create a **ReplicatedMergeTree** table for configuring the TTL for cold and hot storage.

   ClickHouse supports table-level TTLs and allows you to set time-based rules to move data between specified disks or volumes. Data is stored at different storage layers.

   The following is a table creation statement:

```
CREATE TABLE example_table on cluster default_cluster
 (
d DateTime,
a Int
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/example_table', '{replica}')
PARTITION BY toYYYYMM(d)
ORDER BY d
TTL d + INTERVAL 3 YEAR,
d + INTERVAL 1 YEAR TO VOLUME 'cold_obs_volume'
SETTINGS storage_policy = 'hot_cold_separation_policy';
```

   NOTE

   In this statement, the TTL configuration is in bold. This example deletes cold data older than three years from the **example_table** table, and moves cold data older than one year to OBS. The **d** column stores the time information about the data.

   The TTL policy is specified using a simple SQL expression that contains the time and time interval. For example:

   - The data will expire in three days since its creation (**date_time**).
     ```
     TTL date_time + INTERVAL 3 DAY
     ```
   - The data will expire one year later since its creation (**date_time**).
     ```
     TTL date_time + INTERVAL 1 YEAR
     ```

   The INTERVAL keyword supports **second**, **minute**, **hour**, **day**, **week**, **month**, **quarter**, and **year**.

   storage_policy indicates the specified custom storage policy. In this example, the value is the policy name **_clickhouse.storage_configuration.disks** defined in **3**.

Importing Data into ClickHouse to Verify the Configuration

1. Insert data for verification.

```
insert into example_table values('2023-12-27','10086');  -- hot data
insert into example_table values('2023-12-26','10086');  -- hot data
```



```
insert into example_table values('2022-12-24','10086');  -- cold data
insert into example_table values('2022-11-24','10086');  -- cold data
insert into example_table values('2018-10-01','10086');  -- deleted data
insert into example_table values('2017-10-01','10086');  -- deleted data
```

2. Query table data.

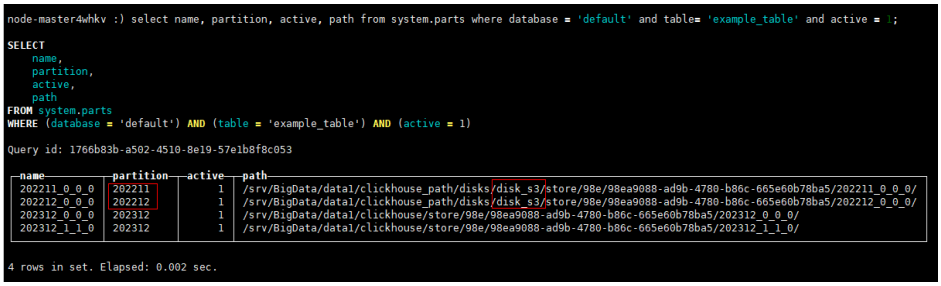
```
tesspmrB0002.mrs-dviw.com :) select * from example_table FORMAT CSV;
SELECT *
FROM example_table
FORMAT CSV
Query id: 3d6bfcb4-f082-4e97-8d95-f885e22ae689

"2023-12-27 00:00:00",10086
"2023-12-26 00:00:00",10086
"2022-11-24 00:00:00",10086
"2022-12-24 00:00:00",10086

4 rows in set. Elapsed: 0.037 sec.
```

Assume that current system time is April 2024. According to the TTL policy rule **example_table**, the data generated on 2018-10-01 and 2017-10-01 will be deleted.

3. Query the storage paths of hot data and cold data.

select name, partition, active, path from system.parts where database ='*database name*'**and table='***table name*' **and active = 1;**

Figure 3-9 Query result



Assume that the current system time is April 2024. Data older than one year (**d** column values: **202211** and **202212**) in the **example_table** table is stored in the **disk_s3** OBS bucket.

3.5.8 Configuring the Connection Between ClickHouse and Open-Source ClickHouse

NOTE

This section applies to MRS 3.3.0-LTS.1 or later.

Scenario

For clusters in normal mode (Kerberos authentication disabled), you need to configure CLICKHOUSE_OPENSOURCE_COMMUNITY to connect to the open-source or other vendors' ClickHouse. For clusters in security mode (Kerberos authentication enabled), the connection is not supported.

Parameter Configuration

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** then **All Configurations**, click **ClickHouseServer(Role)** > **Security**, and search for and modify the following parameters.

Parameter	Description
CLICKHOUSE_OPENSOURCE_COMMUNITY	Whether to support the connection to the open-source ClickHouse. The default value is **false**, indicating that ClickHouse cannot be connected to an open-source ClickHouse. Value **true** indicates that ClickHouse can be connected to an open-source ClickHouse.

Step 2 Click **Save**. In the displayed dialog box, click **OK** to save the configuration. Choose **Instances**, select **ClickHouseServer**, and click **More** > **Instance Rolling Restart**.

----End

3.5.9 Pre-Caching ClickHouse Metadata to the Memory

NOTE

This section applies to MRS 3.3.1-LTS or later.

Scenario

If there are a large number of service tables holding a large amount of data, loading metadata during rolling restart is time-consuming. You can use RocksDB to pre-cache the metadata to the memory to accelerate metadata loading.

Enabling Metadata Pre-Caching

You can set **use_metadata_cache** to **1** or **true** to cache metadata to the memory through RocksDB.

1. Use the ClickHouse client to connect to the ClickHouse server by referring to **ClickHouse Client Practices**.
2. Configure metadata pre-caching.
   - Enable metadata pre-caching for historical tables.

     ALTER TABLE *<table name>* **MODIFY SETTING use_metadata_cache=1;**

     Or

     ALTER TABLE *<table name>* **MODIFY SETTING use_metadata_cache=true;**
   - Enable metadata pre-caching when you create a table.

     CREATE TABLE *<table name>*

     (

     `x` UInt32,

     `y` UInt32,

```
`z` UInt32,
`t` UInt32
)
ENGINE = MergeTree
PARTITION BY x % 10
ORDER BY (x, y)
SETTINGS index_granularity = 8192, use_metadata_cache = 1
```

or

```
CREATE TABLE <table name>
(
`x` UInt32,
`y` UInt32,
`z` UInt32,
`t` UInt32
)
ENGINE = MergeTree
PARTITION BY x % 10
ORDER BY (x, y)
SETTINGS index_granularity = 8192, use_metadata_cache = true
```

Parameter Tuning

Metadata pre-caching can be optimized as follows:

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse**, click **Configurations** and then **All Configurations**, and modify the following parameters:

Parameter	Value	Description
merge_tree_metadata_cache.continue_if_corrupted	true	If the local RocksDB directory fails to be read, **false** indicates that you can exit the process, and **true** indicates that dirty data is cleared.
merge_tree_metadata_cache.lru_cache_size	1GB	Size of the LRU in the RocksDB instance used to cache part metadata

3.6 ClickHouse Performance Tuning

3.6.1 Optimizing ClickHouse Table Partitioning

Troubleshooting

1. Log in to the ClickHouse client and check whether abnormal merge exists.

   select database, table, elapsed, progress, merge_type from system.merges;

2. Do not perform the INSERT operation too frequently, do not insert a small amount of data, and increase the interval for inserting data.
3. The data table partitions are not properly allocated. As a result, too many partitions are generated and data tables need to be re-partitioned.
4. If the MERGE operations are not triggered or slow, adjust the following parameters to accelerate them.

   For details, see **Accelerating ClickHouse Merge**.

Configuration Item	Reference Value
max_threads	Number of CPU cores x 2
background_pool_size	Number of CPU cores
merge_max_block_size	The value is an integer multiple of 8192 and is adjusted based on the CPU and memory resources.
cleanup_delay_period	Set this parameter to a value that is appropriately less than the default value 30.

Changing the Value of parts_to_throw_insert

CAUTION

Increase the value of this parameter only in special scenarios. This configuration acts as a warning for potential issues to some extent. If the cluster hardware resources are insufficient and this configuration is not adjusted properly, potential service issues cannot be detected in a timely manner, which may cause other faults and increase the difficulty of fault recovery.

- For versions earlier MRS 3.2.0, log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** then **All Configurations**. Click **ClickHouseServer** > **Customization**, find the **clickhouse-config-customize** parameter, add the configuration in **Table 3-14**, save it, and restart the service.
- For MRS 3.2.0 and later versions, log in to FusionInsight Manager, choose **Cluster** > **ClickHouse** > **Configurations** > **All Configurations**, search for and change the value of **merge_tree.parts_to_throw_insert**, save the configurations, and restart the service.

Table 3-14 Parameters

Name	Value
merge_tree.parts_to_throw_insert	Memory of ClickHouse instances/32 GB x 300 (conservative estimation)

Verify the modification.

Log in to the ClickHouse client and run the **select * from system.merge_tree_settings where name = 'parts_to_throw_insert';** command.

3.6.2 Accelerating ClickHouse Merge

To accelerate background tasks, adjust the ZooKeeper service configuration first. Otherwise, the ClickHouse service and background tasks will be abnormal due to insufficient ZooKeeper resources such as znodes.

1. Adjust the ZooKeeper configuration. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Zookeeper**. Click **Configurations** then **All Configurations**. Click **quorumpeer** > **System**, change the value of **GC_OPTS** according to the following table, save the configuration, and roll restart the ZooKeeper service.

Configuration Item	Reference Value	Description
GC_OPTS	Xmx = (Memory size of master nodes – 16 GB) x 0.65 (conservative estimation)	JVM parameter used for garbage collection (GC). This parameter is valid only when **GC_PROFILE** is set to **custom**. Ensure that the **GC_OPT** parameter is set correctly. Otherwise, the process will fail to be started. **CAUTION** Exercise caution when modifying this item. If this parameter is set incorrectly, the service will be unavailable.

2. Adjust the ClickHouse configuration. On FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** then **All Configurations**. Click **ClickHouseServer** > **Zookeeper**, modify the following parameters, and save the configuration. You do not need to restart the service.

Configuration Item	Reference Value	Description
clickhouse.zookeeper.quota.node.count	Xmx/4 GB x 1,500,000	Node quantity quota of ClickHouse in the top directory on ZooKeeper. This parameter cannot be set to **0**, but can be set to a minimum value of **–1**. **–1** indicates that there is no limit on the number of ClickHouse nodes in the top directory. **CAUTION** If the quantity quota is less than the actual value of the current ZooKeeper directory, the configuration can be saved but does not take effect and an alarm is reported on the GUI.
clickhouse.zookeeper.quota.size	Xmx/4 GB x 1 GB	Capacity quota of ClickHouse in the top directory on ZooKeeper. **CAUTION** If the quantity quota is less than the actual value of the current ZooKeeper directory, the configuration can be saved but does not take effect and an alarm is reported on the GUI.

3.6.3 Accelerating ClickHouse TTL Operations

When TTL is triggered in ClickHouse, a large amount of CPU and memory are consumed.

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Configurations** then **All Configurations**. Click **ClickHouseServer** > **Customization**, find the **clickhouse-config-customize** parameter, add the following parameters, save the configuration, and restart the service.

Configuration Item	Reference Value	Description
merge_tree.max_replicated_merges_with_ttl_in_queue	Half of number of CPU cores	Number of tasks that allow TTL to merge parts concurrently in the ReplicatedMergeTree queue.
merge_tree.max_number_of_merges_with_ttl_in_pool	Number of CPU cores	The thread pool that allows TTL to merge parts in the ReplicatedMergeTree queue.

NOTE

Do not modify these configurations when the cluster writes heavily. Idle threads need to be reserved for regular Merge operations to avoid the "Too many parts" issue.

3.7 ClickHouse O&M Management

3.7.1 ClickHouse Log Overview

Logs of MRS 3.2.0 and Later Versions

Log path: ClickHouse logs are stored in **${BIGDATA_LOG_HOME}/clickhouse** by default.

- ClickHouse run logs: **/var/log/Bigdata/clickhouse/clickhouseServer/ *.log**
- Balancer run logs: **/var/log/Bigdata/clickhouse/balance/*.log**
- Data migration logs: **/var/log/Bigdata/clickhouse/migration/${task_name}/clickhouse-copier_{timestamp}_{processId}/copier.log**
- ClickHouse audit logs: **/var/log/Bigdata/audit/clickhouse/clickhouse-server.audit.log**

Log archiving rules:

- ClickHouse compresses and archives logs. By default, when the size of a log file exceeds 100 MB, the log file will be compressed.
- The file generated after log files are compressed is named in the format of **<***Original log name***>.[***ID***].gz**.
- A maximum of 10 latest compressed files are reserved by default. The number of compressed files can be configured on Manager.

Table 3-15 ClickHouse log list

Log Type	Log File Name	Description
ClickHouse log	/var/log/Bigdata/clickhouse/clickhouseServer/clickhouse-server.err.log	Path of ClickHouseServer error log files
	/var/log/Bigdata/clickhouse/clickhouseServer/checkService.log	Path of key ClickHouseServer run log files
	/var/log/Bigdata/clickhouse/clickhouseServer/clickhouse-server.log	
	/var/log/Bigdata/clickhouse/clickhouseServer/ugsync.log	User role synchronization tool log
	/var/log/Bigdata/clickhouse/clickhouseServer/prestart.log	ClickHouse prestart log
	/var/log/Bigdata/clickhouse/clickhouseServer/start.log	ClickHouse startup log

Log Type	Log File Name	Description
	/var/log/Bigdata/clickhouse/clickhouseServer/checkServiceHealthCheck.log	ClickHouse health check log
	/var/log/Bigdata/clickhouse/clickhouseServer/checkugsync.log	User role synchronization check log
	/var/log/Bigdata/clickhouse/clickhouseServer/checkDisk.log	Path of ClickHouse disk check log files
	/var/log/Bigdata/clickhouse/clickhouseServer/backup.log	Path of log files generated when ClickHouse performs the backup and restoration operations on Manager
	/var/log/Bigdata/clickhouse/clickhouseServer/stop.log	ClickHouse stop log
	/var/log/Bigdata/clickhouse/clickhouseServer/postinstall.log	postinstall.sh script invoking log of ClickHouse
	/var/log/Bigdata/clickhouse/balance/start.log	Path of ClickHouseBalancer startup log files
	/var/log/Bigdata/clickhouse/balance/error.log	Path of ClickHouseBalancer error log files
	/var/log/Bigdata/clickhouse/balance/access_http.log	Path of the HTTP log files generated during ClickHouseBalancer running
	/var/log/Bigdata/clickhouse/balance/access_tcp.log	Path of the TCP log files generated during ClickHouseBalancer running
	/var/log/Bigdata/clickhouse/balance/checkService.log	ClickHouseBalancer service check log
	/var/log/Bigdata/clickhouse/balance/postinstall.log	Invoking log of the **postinstall.sh** script of ClickHouseBalcer
	/var/log/Bigdata/clickhouse/balance/prestart.log	Path of prestart log files of ClickHouseBalancer
	/var/log/Bigdata/clickhouse/balance/stop.log	Path of stop log files of ClickHouseBalancer
	/var/log/coredump/clickhouse-*.core.gz	Compressed package of memory dump files generated after the ClickHouse process breaks down This log is available in MRS 3.3.0 or later versions only.

Log Type	Log File Name	Description
Data migration log	/var/log/Bigdata/clickhouse/migration/*Data migration task name*/clickhouse-copier_*{timestamp}_{processId}*/copier.log	Run log generated when you use the migration tool by referring to **Migrating Data Between ClickHouseServer Nodes in a Cluster**
	/var/log/Bigdata/clickhouse/migration/*Data migration task name*/clickhouse-copier_*{timestamp}_{processId}*/copier.err.log	Error log generated when you use the migration tool by referring to **Migrating Data Between ClickHouseServer Nodes in a Cluster**
clickhouse-tomcat log	/var/log/Bigdata/tomcat/clickhouse/web_clickhouse.log	ClickHouse custom UI run log
	/var/log/Bigdata/tomcat/audit/clickhouse/clickhouse_web_audit.log	Clickhouse data migration audit log
ClickHouse audit log	/var/log/Bigdata/audit/clickhouse/clickhouse-server-audit.log	Path of ClickHouse audit log files

Logs of MRS 3.2.0 and Earlier Versions

Log path: The default storage path of ClickHouse log files is as follows: **${BIGDATA_LOG_HOME}/clickhouse**

Log archive rule: The automatic log compression and archiving function are enabled. By default, when the size of logs exceeds 100 MB, logs are compressed into a log file named in the format of *<Original log file name>.[No.]***.gz**. A maximum of 10 latest compressed files are reserved by default. The number of compressed files can be configured on Manager.

Table 3-16 ClickHouse log list

Log Type	Log File Name	Description
Run log	/var/log/Bigdata/clickhouse/clickhouseServer/clickhouse-server.err.log	Path of ClickHouseServer error log files
	/var/log/Bigdata/clickhouse/clickhouseServer/checkService.log	Path of key ClickHouseServer run log files
	/var/log/Bigdata/clickhouse/clickhouseServer/clickhouse-server.log	
	/var/log/Bigdata/clickhouse/balance/start.log	Path of ClickHouseBalancer startup log files
	/var/log/Bigdata/clickhouse/balance/error.log	Path of ClickHouseBalancer error log files

Log Type	Log File Name	Description
	/var/log/Bigdata/clickhouse/balance/access_http.log	Path of ClickHouseBalancer run log files
Data migration log	/var/log/Bigdata/clickhouse/migration/*Data migration task name*/clickhouse-copier_{timestamp}_{processId}/copier.log	Run log generated when you use the migration tool by referring to **Migrating Data Between ClickHouseServer Nodes in a Cluster**
	/var/log/Bigdata/clickhouse/migration/*Data migration task name*/clickhouse-copier_{timestamp}_{processId}/copier.err.log	Error log generated when you use the migration tool by referring to **Migrating Data Between ClickHouseServer Nodes in a Cluster**

Log Level

Table 3-17 describes the log levels supported by ClickHouse.

Levels of run logs are error, warning, trace, information, and debug from the highest to the lowest priority. Run logs of equal or higher levels are recorded. The higher the specified log level, the fewer the logs recorded.

Table 3-17 Log levels

Level	Description
error	Logs of this level record error information about system running
warning	Logs of this level record exception information about the current event processing
trace	Logs of this level record trace information about the current event processing
information	Logs of this level record normal running status information about the system and events
debug	Logs of this level record system running and debugging information

To modify log levels, perform the following operations:

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **ClickHouse** > **Configurations**.

Step 3 Select **All Configurations**.

Step 4 On the menu bar on the left, select the log menu of the target role.

Step 5 Select a desired log level.

Step 6 Click **Save**. Then, click **OK**.

----End

NOTE

The configurations take effect immediately without the need to restart the service.

Log Format

The following table lists the ClickHouse log format:

Table 3-18 Log formats

Log Type	Format	Example
ClickHouse run log	<*yyyy-MM-dd HH:mm:ss,SSS*>|<*Log level*>|<*Name of the thread that generates the log*>|<*Message in the log*>|<*Location where the log event occurs*>	2021.02.23 15:26:30.691301 [6085] {} <Error> DynamicQueryHandler: Code: 516, e.displayText() = DB::Exception: default: Authentication failed: password is incorrect or there is no user with such name, Stack trace (when copying this message, always include the lines below): 0. Poco::Exception::Exception(std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> > const&, int) @ 0x1250e59c
clickhouse-tomcat run log	<*yyyy-MM-dd HH:mm:ss,SSS*>|<*Log level*>|<*Name of the thread that generates the log*>|<*Message in the log*>|<*Location where the log event occurs*>	2022-08-16 12:55:12,109 | INFO | pool-7-thread-1 | zookeeper is secure. | com.huawei.bigdata.om.extui.clickhouse.service.impl.QueryServiceImpl.initAuthContext(QueryServiceImpl.java:136)
Data migration log	<*yyyy-MM-dd HH:mm:ss,SSS*>|<*Log level*>|<*Name of the thread that generates the log*>|<*Message in the log*>|<*Location where the log event occurs*>	2022.08.07 14:41:01.814235 [28651] {} <Debug> ClusterCopier: Task / clickhouse/copier_tasks/TEST0807_02/tables/dblv85.startsea_zh_imoriginck_new/20201031/piece_4/shards/1 has been successfully executed by 8%2D5%2D226%2D156#20220807124849_28651

Log Type	Format	Example
Audit log	<*yyyy-MM-dd HH:mm:ss,SSS*>|*query id*|<*Log Level*><*Name of the thread that generates the log*>|<*Message in the log*>|<*Location where the log event occurs*>	2022.08.16 20:58:16.723643 [11382] {cc9554b6-8a26-42e9-8ab8-d848500544e6} <Information> executeQuery_audit [executeQuery.cpp:202] : (0 from 192.168.64.81:45204, user: clickhouse, using experimental parser) select shard_num, host_name, host_address from system.clusters format JSON

3.7.2 Collecting Dumping Logs of the ClickHouse System Tables

NOTE

This topic is available for MRS 3.3.0-LTS and later versions only.

Scenario

If an exception occurs, you need to restart ClickHouse to restore services. Before restart, you need to dump the status information of each ClickHouse system table for efficient ClickHouse fault locating.

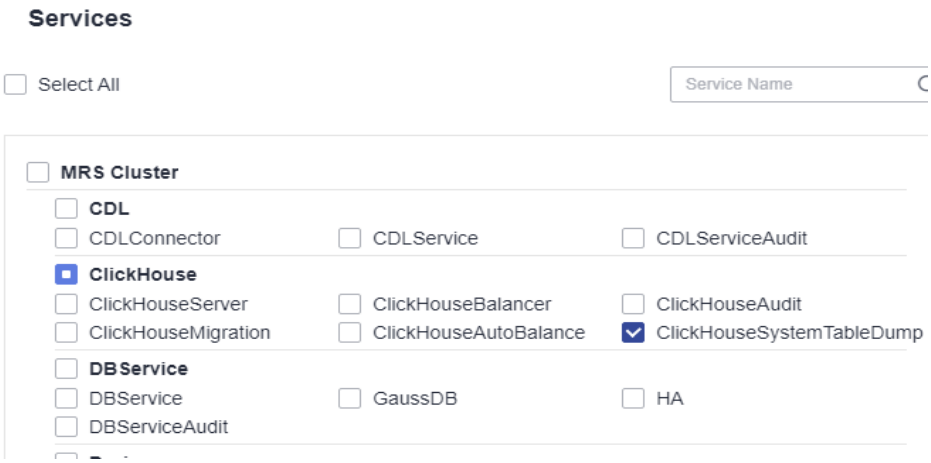
System table logs can be dumped in real time or in one-click mode, as shown in the following table.

Log Dumping Type	System Tables
Real-time dumping	• system.asynchronous_metrics • system.clusters • system.distribution_queue • system.events • system.grants • system.mutations • system.processes • system.metrics • system.part_moves_between_shards • system.replicas • system.replicated_fetches • system.replication_queue

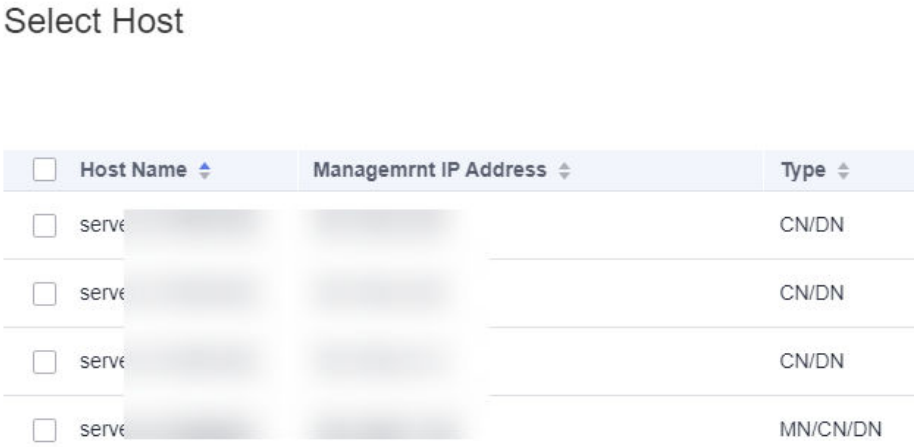
Log Dumping Type	System Tables
One-click dumping	• system.distributed_ddl_queue • system.errors • system.parts • system.parts_columns • system.query_log • system.query_thread_log • system.trace_log

Collecting Real-Time Dumping Logs

Step 1 Log in to FusionInsight Manager, choose **O&M** > **Log** > **Download**, and select **ClickHouseSystemTableDump** for **Service**.



Step 2 Select the hosts where you want to collect dumping logs and click **OK**.



Step 3 Click the time editing button in the upper right corner and set **Start Time** and **End Time** for log collection.

NOTE

For details about how long it takes to collect logs, contact technical support engineers.

Step 4 Click **Download**. The real-time system table dumping logs are saved to the local PC.

----End

Collecting One-Click Dumping Logs

Step 1 Log in to any ClickHouseServer node as the **root** user and go to the **sbin** directory.

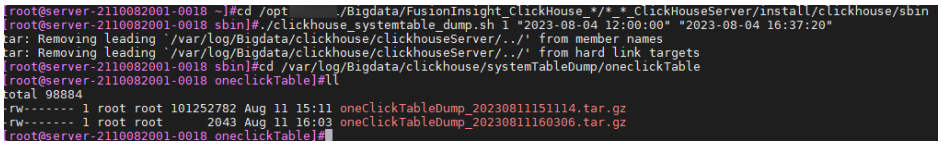
cd ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/*_*_ClickHouseServer/install/clickhouse/sbin

Step 2 Run the following command to obtain dumping logs:

./clickhouse_systemtable_dump.sh 1 "*Start time***" "***End time***"**

Example: **./clickhouse_systemtable_dump.sh 1 "2023-08-04 12:00:00" "2023-08-04 16:37:20"**

Step 3 Go to the **/var/log/Bigdata/clickhouse/systemTableDump/oneclickTable** directory and view the compressed one-click dumping logs.



----End

3.7.3 Enabling the Read-Only Mode for ClickHouse Tables

NOTE

This topic is available for MRS 3.2.0 or later only.

Scenario

During data migration, one-click balancing, decommissioning and capacity reduction, ClickHouse allows you to set the **only_allow_select_statement** parameter for MergeTree series tables to enable SELECT operations instead of ALTER, RENAME, DROP, and INSERT operations.

Procedure for Enabling the Read-Only Mode of the ClickHouse Table

Step 1 Install the client. For details, see **Installing a Client**.

Step 2 Run the following commands to log in to the node where the client is installed as user **root**:

cd *Client installation directory*

source bigdata_env

Step 3 Run the following command to authenticate the user if the cluster is in security mode (with Kerberos authentication enabled). Otherwise, skip this step.

kinit *Component service user*

NOTE

The user must have the ClickHouse administrator permissions.

Step 4 Run the proper client command to connect to the ClickHouse server.

- Normal mode

  clickhouse client --host *IP address of the ClickHouse instance* **--user** *Username* **--password --port** 9440 **--secure**

  Enter the user password.

- Security mode

  clickhouse client --host *IP address of the ClickHouse instance* **--port** 9440 **--secure**

NOTE

- For a cluster in normal mode, the username is **default**. Alternatively, you can create an administrator using the open source ClickHouse web UI. Do not use the users created on FusionInsight Manager.
- To obtain the IP address of the ClickHouseServer instance, log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse**, and click the **Instance** tab.

Step 5 Run the following statement to set the table to read-only:

ALTER TABLE *{table_name}* **MODIFY SETTING only_allow_select_statement = true;**

----End

Disabling the Read-Only Mode of the Table

Step 1 Log in to the ClickHouse client by referring to **Step 2** to **Step 4**.

Step 2 Run the following statement to disable the read-only mode of the table:

ALTER TABLE *{table_name}* **MODIFY SETTING only_allow_select_statement = false settings hw_internal_operation = true;**

----End

3.7.4 Migrating Data Between ClickHouseServer Nodes in a Cluster

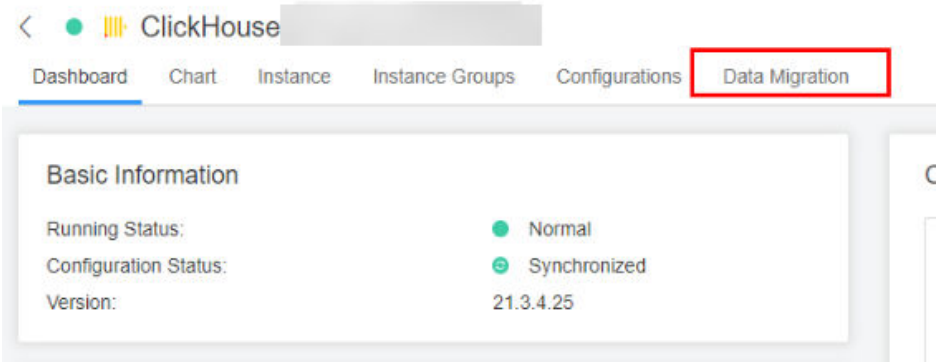
The ClickHouse data migration tool can migrate some partitions of one or more partitioned MergeTree tables on several ClickHouseServer nodes to the same tables on other ClickHouseServer nodes. In the capacity expansion scenario, you can use this tool to migrate data from an original node to a new node to balance data after capacity expansion.

Prerequisites

- The ClickHouse and Zookeeper services are running properly. The ClickHouseServer instances on the source and destination nodes are normal.
- The destination node has the data table to be migrated and the table is a partitioned MergeTree table.
- Before creating a migration task, ensure that all tasks for writing data to a table to be migrated have been stopped. After the task is started, you can only query the table to be migrated and cannot write data to or delete data from the table. Otherwise, data may be inconsistent before and after the migration.
- The ClickHouse data directory on the destination node has sufficient space.

Procedure

Step 1 Log in to Manager and choose **Cluster** > **Services** > **ClickHouse**. On the ClickHouse service page, click the **Data Migration** tab.



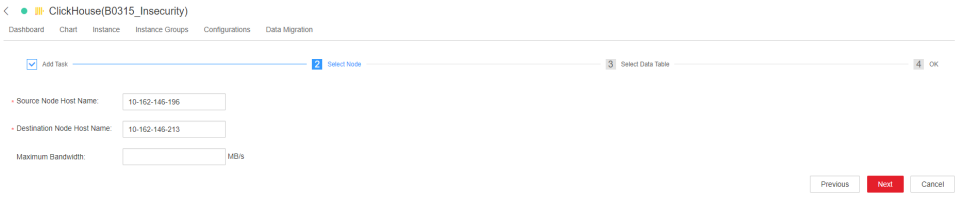
Step 2 Click **Add Task**.

Step 3 On the page for creating a migration task, set the migration task parameters. For details, see **Table 3-19**.

Table 3-19 Migration task parameters

Parameter	Description
Task Name	Enter a specific task name. The value can contain 1 to 50 characters, including letters, arrays, and underscores (_), and cannot be the same as that of an existing migration task.
Task Type	• **Scheduled Task**: When the scheduled task is selected, you can set **Started** to specify a time point later than the current time to execute the task. • **Immediate task**: The task is executed immediately after it is started.
Started	Set this parameter when **Task Type** is set to **Scheduled Task**. The valid value is a time point within 90 days from now.

Step 4 On the **Select Node** page, specify **Source Node Host Name** and **Destination Node Host Name**, and click **Next**.

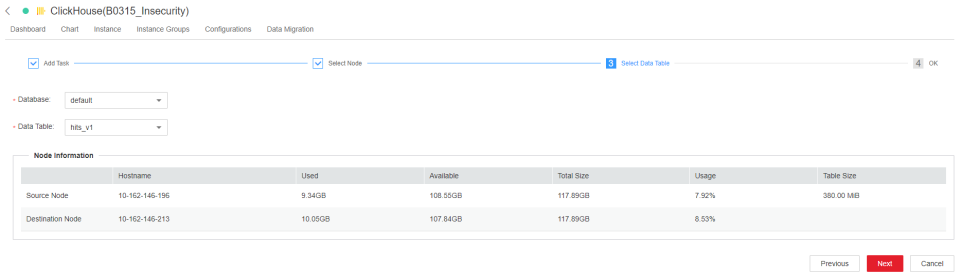


NOTE

- Only one host name can be entered in **Source Node Host Name** and **Destination Node Host Name**, respectively. Multi-node migration is not supported.

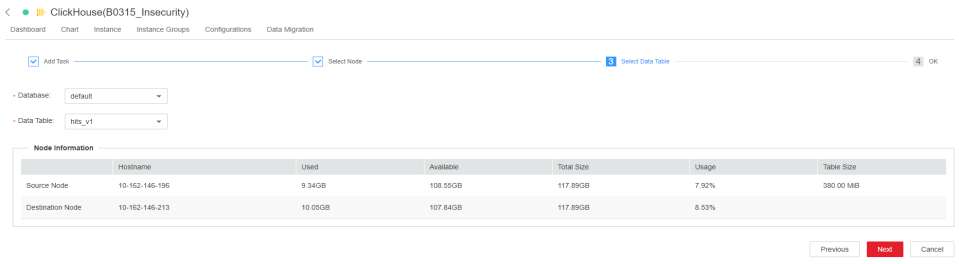
  To obtain the parameter values, click the **Instance** tab on the ClickHouse service page and view the **Host Name** column of the current ClickHouseServer instance.
- **Maximum Bandwidth** is optional. If it is not specified, there is no upper limit. The maximum bandwidth can be set to **10000** MB/s.

Step 5 On the **Select Data Table** page, click next to **Database**, select the database to be migrated on the source node, and select the data table to be migrated for **Data Table**. The data table drop-down list displays the partitioned MergeTree tables in the selected database. In the **Node Information** area, the space usage of the ClickHouse service data directory on the current source and destination nodes is displayed. Click **Next**.



Step 6 Confirm the task information and click **Submit**.

The data migration tool automatically calculates the partitions to be migrated based on the size of the data table. The amount of data to be migrated is the total size of the partitions to be migrated.



Step 7 After the migration task is submitted, click **Start** in the **Operation** column. If the task is an immediate task, the task starts to be executed. If the task is a scheduled task, the countdown starts.

Step 8 During the migration task execution, you can click **Cancel** to cancel the migration task that is being executed. If you cancel the task, the migrated data on the destination node will be rolled back.

You can choose **More** > **Details** to view the log information during the migration.

Step 9 After the migration is complete, choose **More** > **Results** to view the migration result and choose **More** > **Delete** to delete the directories related to the migration task on ZooKeeper and the source node.

----End

3.7.5 Migrating ClickHouse Data from One MRS Cluster to Another

NOTE

This section applies only to MRS 3.2.0 or later.

Scenario

Scenario 1: As the number of MRS ClickHouse services increases, the storage and compute resources of clusters cannot meet service requirements. Thus, the clusters need to be split so that part of user service and database data can be migrated to new clusters.

Scenario 2: The data center where the backend hosts of MRS ClickHouse clusters are located needs to be migrated, so does the ClickHouse cluster data.

To meet the migration requirements, MRS provides a one-click data migration tool for migrating the databases, table objects (DDL), and service data of ClickHouse from a source cluster to the new cluster.

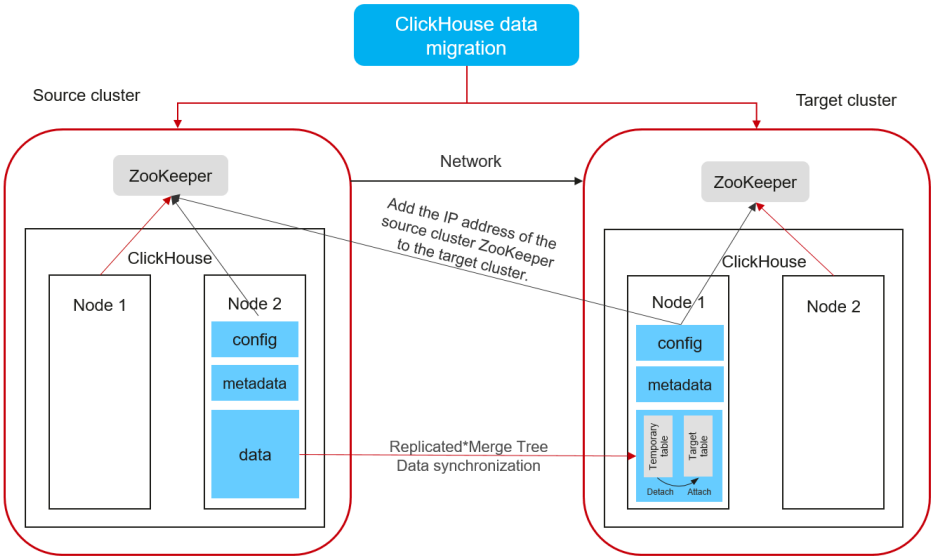
Migration Mechanism

- Replicated*MergeTree table migration

  In this migration solution, ClickHouse uses ZooKeeper to automatically synchronize the data of Replicated*MergeTree tables of different replicas in the same shard. The logical procedure is as follows:

  First, add the ZooKeeper information of the source cluster to the configuration file of the destination cluster as the auxiliary ZooKeeper. Then, in the destination cluster, create a temporary table that has the same ZooKeeper path and structure as the source cluster but different replicas from it. After the temporary table is created, data in the source cluster will be automatically synchronized to the temporary table. Once data synchronization is complete, copy data from the temporary table to the formal table.

Figure 3-10 Replicated*MergeTree table migration architecture



- Distributed table migration

  During the migration of a replicated table, its metadata in the source cluster is exported and changed to the ZooKeeper path and replica of the destination cluster. Then, you can create a table in the destination cluster based on the modified metadata.

- Non-replicated table and materialized view migration

  To migrate data in the non-replicated tables and materialized views, you can call the **remote** function.

The preceding migration operations are encapsulated using the migration tool script. In this way, you can modify the related configuration files and run the migration scripts to complete the migration by one click. For details, see the procedure description.

Prerequisites

- The status of the ClickHouse cluster you want to migrate is normal, and the source and destination clusters must be both in security mode or normal mode. If the clusters are in normal mode, contact O&M personnel for migration.
- A destination ClickHouse cluster of MRS 3.1.3 or later has been created for data migration. The cluster must be in security mode. The number of ClickHouseserver instances in the ClickHouse cluster is greater than or equal to that of the source cluster.
- Currently, logical clusters support only data migration to clusters with the same number of replicas.

Constraints

- Only table data and table object metadata (DDL) can be migrated. SQL statements like ETL need to be manually migrated.
- To ensure data consistency before and after the migration, stop the ClickHouse service of the source cluster before the migration. For details, see the procedure description.

- If the table of the source cluster is deleted during the migration, this issue can only be solved manually.

Procedure

The overall migration procedure is as follows:

Figure 3-11 Migration flowchart

Table 3-20 Migration description

Step	Description
Step 1: Connect the source cluster to the destination cluster.	This step ensures that the source and target ClickHouse clusters as well as their nodes can communicate with each other.
Step 2: Add the ZooKeeper information of the source cluster to the configuration file of the destination cluster.	By doing so, ZooKeeper in the source cluster can be used as the auxiliary ZooKeeper during data migration.
Step 3: Migrate metadata of databases and tables in the source ClickHouse cluster to the destination cluster.	You can run the corresponding script to migrate metadata such as the database name, table name, and table structure of the ClickHouse database and tables in the source cluster to the destination cluster.
Step 4: Migrate data of the databases and tables in the source ClickHouse cluster to the destination cluster.	You can run the corresponding script to migrate the ClickHouse database and table data from the source cluster to the destination cluster.

Connecting the Source Cluster to the Destination Cluster

1. Connect the source cluster to the destination cluster so that the ClickHouse instance nodes in the two clusters can communicate with each other.
2. Add the hosts information of the source cluster to the configuration of all nodes in the destination cluster, and add the hosts information of the destination cluster to the configuration of all nodes in the source cluster.
   a. Log in to FusionInsight Manager of the source ClickHouse cluster, choose **Cluster** > **ClickHouse**, click the **Instance** tab, and view the service IP address of the ClickHouseServer instance node.

b. Log in to any ClickHouseServer node using SSH and run the following command to check the host configurations of the ClickHouse instance in the source cluster:

cat /etc/hosts

The following figure shows the host configurations of the ClickHouse instance:

```
[root@19                ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.64.59 100-93-30-239 100-93-30-239.
192.168.64.157 100-95-140-144 100-95-140-144.
192.168.64.139 100-94-12-218 100-94-12-218.
192.168.64.81 100-94-163-99 100-94-163-99.
192.168.64.66 192-168-64-66 192-168-64-66.
192.168.64.24 192-168-64-24 192-168-64-24.
192.168.64.91 192-168-64-91 192-168-64-91.
192.168.64.162 192-168-64-162 192-168-64-162.
10.10.10.10 hadoop.hadoop.com
```

c. Log in to FusionInsight Manager of the destination ClickHouse cluster, choose **Cluster** > **ClickHouse**, click the **Instance** tab, and view the service IP address of the ClickHouseServer instance node in the destination cluster.

d. Log in to all ClickHouse nodes in the destination cluster as user **root** and run the following command to modify the **/etc/hosts** configuration of the nodes:

vi /etc/hosts

Copy the host information of the ClickHouse instance of the source cluster obtained in **2.b** to the **hosts** file.

e. Repeat **2.a** to **2.d** to add the node IP addresses of the destination cluster to the hosts files of the source cluster.

3. Configure system mutual trust between the source cluster and the destination cluster. For details, see **Configuring Cross-Manager Mutual Trust Between Clusters**.

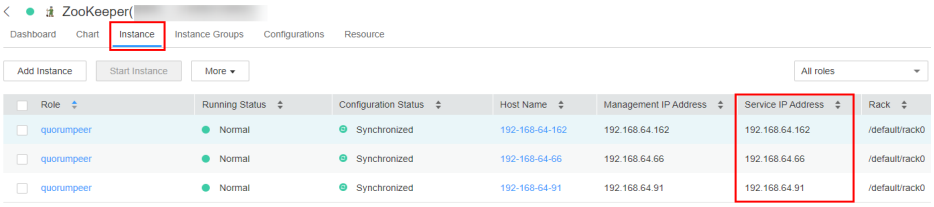
NOTE

If both the source and destination clusters are in normal mode, you do not need to configure mutual trust.

Adding the ZooKeeper Information of the Source Cluster to the Configuration File of the Destination Cluster

1. Log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ZooKeeper**, and click the **Instance** tab. On the displayed page, view the service IP addresses of the ZooKeeper quorumpeer instance , as shown in **Figure 3-12**.

Figure 3-12 Addresses of the source Zookeeper quorumpeer instance



2. Log in to FusionInsight Manager of the destination cluster, choose **Cluster** > **Services** > **ClickHouse** and click the **Configurations** tab and then **All Configurations**. On the displayed page, search for the **clickhouse-config-customize** parameter.
3. Add ZooKeeper instance information of the source cluster to the **clickhouse-config-customize** parameter by referring to **Table 3-21**.

Table 3-21 Configurations of **clickhouse-config-customize**

Parameter	Value
auxiliary_zookeepers.zookeeper2.node[1].host	Service IP address of the first ZooKeeper quorumpeer instance in the source cluster obtained in **1**. Currently, only the IP address of the ZooKeeper instance can be configured.
auxiliary_zookeepers.zookeeper2.node[1].port	2181
auxiliary_zookeepers.zookeeper2.node[2].host	Service IP address of the second ZooKeeper quorumpeer instance in the source cluster obtained in **1**. Currently, only the IP address of the ZooKeeper instance can be configured.
auxiliary_zookeepers.zookeeper2.node[2].port	2181
auxiliary_zookeepers.zookeeper2.node[3].host	Service IP address of the third ZooKeeper quorumpeer instance in the source cluster obtained in **1**. Currently, only the IP address of the ZooKeeper instance can be configured.
auxiliary_zookeepers.zookeeper2.node[3].port	2181

4. After the configuration is complete, click **Save**. In the displayed dialog box, click **OK**.
5. Log in to any ClickHouseServer node of the destination cluster as user **root**. Run the following command to view the ClickHouseServer instance information:

   ps -ef |grep clickhouse

   Obtain the value of **--config-file**, that is, the configuration file directory of the ClickHouseServer, from the query result.

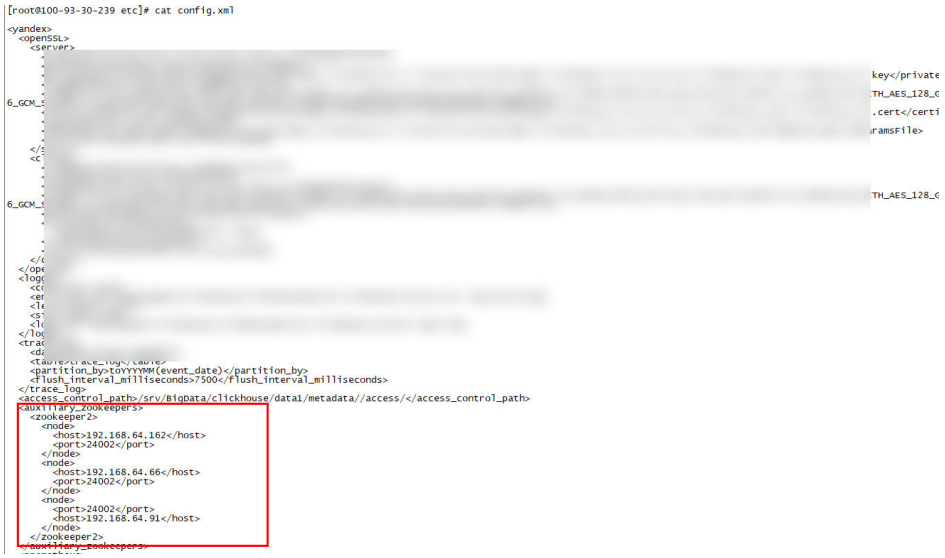
Figure 3-13 Obtaining the configuration file directory of ClickHouseServer

6. Run the corresponding command to check whether the information about **<auxiliary_zookeepers>** is added in the ClickHouse configuration file **config.xml**.

   cat *Directory of the config.xml file obtained in 5*

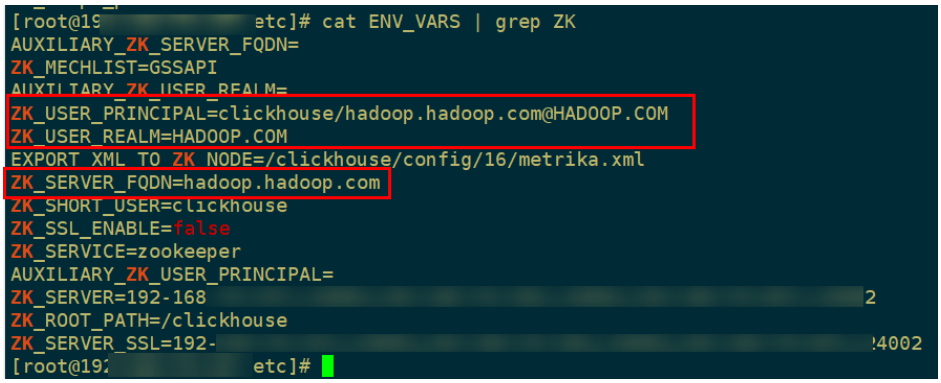
- If both the source and destination clusters are in security mode, go to **7**.
- If both the source and destination clusters are in normal mode, go to **11**.

Figure 3-14 Viewing the added ZooKeeper information of the source cluster



7. In the configuration file directory obtained in **5**, run the following command to obtain the ZooKeeper authentication information of the source cluster:

   cat ENV_VARS | grep ZK



   Obtain the values of **ZK_SERVER_FQDN**, **ZK_USER_PRINCIPAL** and **ZK_USER_REALM**.

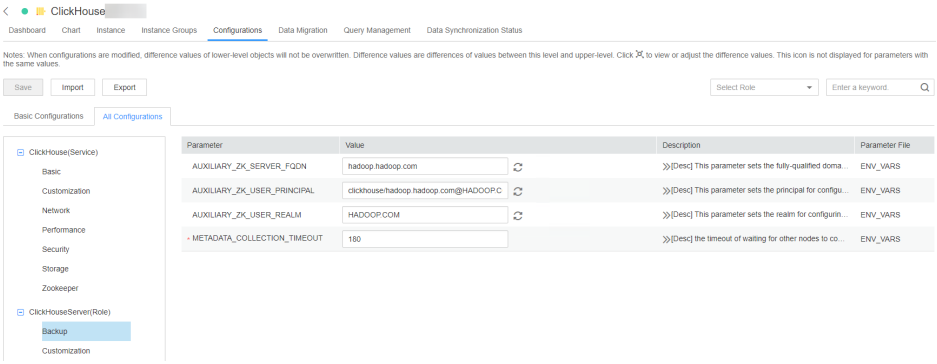
8. Log in to FusionInsight Manager of the destination cluster, choose **Cluster** > **Services** > **ClickHouse**, click the **Configurations** tab and then **All Configurations**. In the navigation pane on the left, choose **ClickHouseServer(Role)** > **Backup** and set the parameters by referring to the following table.

Table 3-22 Configuring the cluster authentication information

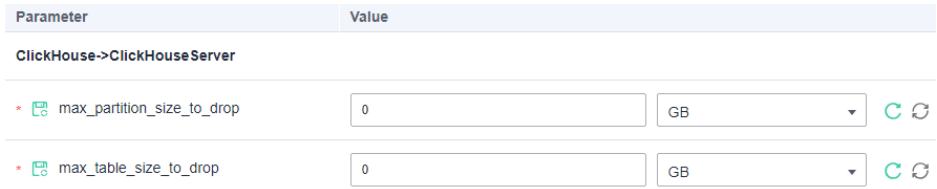
Parameter	Value
AUXILIARY_ZK_SERVER_FQDN	Value of **ZK_SERVER_FQDN** obtained in **7**

Parameter	Value
AUXILIARY_ZK_SERVER_PRINCIPAL	Value of **ZK_USER_PRINCIPAL** obtained in **7**
AUXILIARY_ZK_SERVER_REALM	Value of **ZK_USER_REALM** obtained in **7**
METADATA_COLLECTION_TIMEOUT	**180**. This parameter specifies the timeout interval for waiting for the completion of metadata backup on other nodes, in seconds.

Figure 3-15 Configuring the cluster authentication information



9. Select **Storage** under **ClickHouseServer(Role)**, and change the value of **max_partition_size_to_drop** and **max_table_size_to_drop** to **0**.



10. Click **Save**. In the dialog box that is displayed, click **OK**.
11. On the ClickHouse service page, click the **Instance** tab. On this tab page, select the ClickHouseServer instance from the instance list, and choose **More** > **Restart Instance** in the **Operation** column to restart the ClickHouseServer instance.

Migrating Metadata of Databases and Tables in the Source ClickHouse Cluster to the Destination Cluster

1. Log in to FusionInsight Manager of the source and destination clusters, and create the username and password required for the migration. The procedure is as follows:
   a. Log in to Manager and choose **System** > **Permission** > **Role**. On the displayed page, click **Create Role**.

b. Specify **Role Name**, for example, **ckrole**. In the **Configure Resource Permission** area, click the cluster name. On the displayed service list page, click the ClickHouse service.

c. Select **SUPER_USER_GROUP** and click **OK**.

d. Choose **System**. On the navigation pane on the left, choose **Permission** > **User** and click **Create**.

e. Select **Human-Machine** for **User Type** and set **Password** and **Confirm Password** to the password of the user.

NOTE

- Username: The username cannot contain hyphens (-). Otherwise, the authentication will fail.
- Password: The password cannot contain special characters $, ., and #. Otherwise, the authentication will fail.

f. In the **Role** area, click **Add** . In the displayed dialog box, select the role name in **1.b** and click **OK** to add the role. Then, click **OK**.

g. After the user is created, click the user name in the upper right corner to log out of the system. Log in to FusionInsight Manager as the new user and change its password as prompted.

2. Download the ClickHouse client and install it as user **omm** to the destination cluster.

3. Log in to the client node as user **omm**, go to the *Client installation directory*/**ClickHouse/clickhouse_migration_tool/clickhouse-metadata-migration** directory, and configure migration information. Run the following command to modify the **example_config.yaml** configuration file by referring to **Table 3-23**:

cd *Client installation directory*/**ClickHouse/clickhouse_migration_tool/clickhouse-metadata-migration**

vi example_config.yaml

After the configuration is modified, you must delete all comment with number sign(#) and retain only valid configurations. Otherwise, an error may occur during script migration.

Table 3-23 Parameters in **the example_config.yaml** file

Configuration Item	Sub-item	Value and Description
source_cluster	host	IP address of any ClickHouseServer node in the source cluster.

Configuration Item	Sub-item	Value and Description
	cluster_name	Name of the source ClickHouse cluster. You can log in to the ClickHouse client by referring to **ClickHouse Client Practices** and run the following command to obtain the value. If the source cluster name has not been changed, the default value is **default_cluster**. **select cluster,shard_num,replica_num,host_name from system.clusters;**
	https_port	• For security mode, log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ClickHouse** > **Configurations** > **All Configurations**, and search for **https_port**. • For normal mode, log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ClickHouse** > **Configurations** > **All Configurations**, and search for **http_port**.
	zookeeper_root_path	To obtain the value, log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ClickHouse**, and click the **Configurations** tab and then **All Configurations**. In the displayed page, search for **clickhouse.zookeeper.root.path**.
	system	System parameter. Retain the default value.

	databases	Optional.
		• If this parameter is specified, data in the specified database of the source ClickHouse cluster is migrated. You can specify multiple databases. The following configuration is for your reference:
		`databases:`
		`  - "database"`
		`  - "database_1"`
		Data in the **database** and **database_1** databases of the source cluster is migrated.
		• If this parameter is not specified, table data of all databases in the source ClickHouse cluster is migrated. Leave the **databases** parameter empty. The following is an example:
		`databases:`
		Table information of all databases in the source ClickHouse cluster is migrated.

Configuration Item	Sub-item	Value and Description
	tables	Optional. The value is in the format of *Database name.Table name*. The database name must be in the databases parameter list. • If this parameter is specified, data in specified tables in the source ClickHouse cluster database is migrated. You can configure multiple tables. The following configuration is for your reference: tables: - "database.table_1" - "database_1.table_2" Data in **table_1** of **database** and **table_2** of database_1 of the source cluster is migrated. • If this parameter is not specified and the **databases** parameter is specified, all table data in the **databases** database is migrated. If the **databases** parameter is not specified, all table data in all databases of the source ClickHouse cluster is migrated. The following configuration is for your reference: tables:
destination_cluster	host	IP address of any ClickHouseServer node in the destination cluster.
	cluster_name	Name of the destination ClickHouse cluster. You can log in to the ClickHouse client by referring to **ClickHouse Client Practices** and run the following command to obtain the value. If the destination cluster name has not been changed, the default value is **default_cluster**. **select cluster,shard_num,replica_num,host_name from system.clusters;**
	user	Username created in **1** for logging in to FusionInsight Manager of the destination ClickHouse cluster.
	https_port	• For security mode, log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ClickHouse** > **Configurations** > **All Configurations**, and search for **https_port**. • For normal mode, log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ClickHouse** > **Configurations** > **All Configurations**, and search for **http_port**.

Configuration Item	Sub-item	Value and Description
	zookeeper_root_path	To obtain the value, log in to FusionInsight Manager of the destination cluster, choose **Cluster** > **Services** > **ClickHouse**, and click the **Configurations** tab and then **All Configurations**. In the displayed page, search for **clickhouse.zookeeper.root.path**.
	system	System parameter. Retain the default value.

4. Run the following command to migrate data and wait until the script execution is complete:

   ./clickhouse_migrate_metadata.sh -f yaml_file

   Enter the usernames and passwords of the source and destination clusters.

   ```
   please input source cluster user name:
   please input source cluster user password:
   please input destination cluster user name:
   please input destination cluster user password:
   ```

   NOTE

   If metadata migration fails, perform the following steps:

   1. Locate the failure cause. Specifically, check whether any parameters in the configuration file are incorrectly configured.
      - If yes, reconfigure the parameters and perform metadata migration.
      - If no, go to 2.
   2. Set the names of the tables that fail to be migrated in the metadata migration configuration file based on the **databases** and **tables** parameters in Table 3-23 and run the metadata migration command again. If the migration fails, contact O&M personnel.

Migrating Data of the Databases and Tables in the Source ClickHouse Cluster to the Destination Cluster

1. Log in to the ClickHouse client node in the destination cluster as user **omm** and go to *Client installation directory*/**ClickHouse/clickhouse_migration_tool/clickhouse-data-migration**.

   cd *Client installation directory*/**ClickHouse/clickhouse_migration_tool/clickhouse-data-migration**

2. Run the following command to modify the **example_config.yaml** configuration file by referring to Table 3-24:

   vi example_config.yaml

   After the configuration is modified, you must delete all comment with number sign(#) and retain only valid configurations. Otherwise, an error may occur during script migration.

Table 3-24 Parameters in **example_config.yaml**

Configuration Item	Sub-item	Value and Description
source_cluster	host	IP address of any ClickHouseServer node in the source cluster.
	cluster_name	Name of the source ClickHouse cluster. You can log in to the ClickHouse client by referring to **ClickHouse Client Practices** and run the following command to obtain the value. If the source cluster name has not been changed, the default value is **default_cluster**. **select cluster,shard_num,replica_num,host_name from system.clusters;**
	user	Username created in **1** for logging in to FusionInsight Manager of the source ClickHouse cluster.
	https_port	• For security mode, log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ClickHouse** > **Configurations** > **All Configurations**, and search for **https_port**. • For normal mode, log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ClickHouse** > **Configurations** > **All Configurations**, and search for **http_port**.
	tcp_port	To obtain the value, log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ClickHouse**, and click the **Configurations** tab and then **All Configurations**. In the displayed page, search for **tcp_port_secure** if the cluster is in security mode. Otherwise, search for **tcp_port**.
	zookeeper_root_path	To obtain the value, log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ClickHouse**, and click the **Configurations** tab and then **All Configurations**. In the displayed page, search for **clickhouse.zookeeper.root.path**.
	system	System parameter. Retain the default value.

Configuration Item	Sub-item	Value and Description
	databases	Optional. ● If this parameter is specified, data in the specified database of the source ClickHouse cluster is migrated. You can specify multiple databases. The following configuration is for your reference: databases: - "database" - "database_1" Data in the **database** and **database_1** databases of the source cluster is migrated. ● If this parameter is not specified, table data of all databases in the source ClickHouse cluster is migrated. Leave the **databases** parameter empty. The following is an example: databases: Table information of all databases in the source ClickHouse cluster is migrated.
	tables	Optional. The value is in the format of *Database name*.*Table name*. The database name must be in the databases parameter list. ● If this parameter is specified, data in specified tables in the source ClickHouse cluster database is migrated. You can configure multiple tables. The following configuration is for your reference: tables: - "database.table_1" - "database_1.table_2" Data in **table_1** of **database** and **table_2** of database_1 of the source cluster is migrated. ● If this parameter is not specified and the **databases** parameter is specified, all table data in the **databases** database is migrated. If the **databases** parameter is not specified, all table data in all databases of the source ClickHouse cluster is migrated. The following configuration is for your reference: tables:
destination_cluster	host	IP address of any ClickHouseServer node in the destination cluster.

Configuration Item	Sub-item	Value and Description
	cluster_name	Name of the destination ClickHouse cluster. You can log in to the ClickHouse client by referring to **ClickHouse Client Practices** and run the following command to obtain the value. If the destination cluster name has not been changed, the default value is **default_cluster**. **select cluster,shard_num,replica_num,host_name from system.clusters;**
	user	Username created in **1** for logging in to FusionInsight Manager of the destination ClickHouse cluster.
	https_port	• For security mode, log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ClickHouse** > **Configurations** > **All Configurations**, and search for **https_port**. • For normal mode, log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ClickHouse** > **Configurations** > **All Configurations**, and search for **http_port**.
	tcp_port	To obtain the value, log in to FusionInsight Manager of the destination cluster, choose **Cluster** > **Services** > **ClickHouse**, and click the **Configurations** tab and then **All Configurations**. In the displayed page, search for **tcp_port_secure** if the cluster is in security mode. Otherwise, search for **tcp_port**.
	zookeeper_root_path	To obtain the value, log in to FusionInsight Manager of the destination cluster, choose **Cluster** > **Services** > **ClickHouse**, and click the **Configurations** tab and then **All Configurations**. In the displayed page, search for **clickhouse.zookeeper.root.path**.
	system	System parameter. Retain the default value.
auxiliary_zookeepers	name	ZooKeeper name of the source ClickHouse cluster configured in **3**, for example, **zookeeper2**.

Configuration Item	Sub-item	Value and Description
	hosts	IP address of the ZooKeeper instance of the source ClickHouse. To obtain the IP address, log in to FusionInsight Manager of the source cluster, choose **Cluster** > **Services** > **ZooKeeper**, and click the **Instance** tab. On the displayed page, view the service IP addresses of the ZooKeeper quorumpeer instance , as shown in **Figure 3-12**. The format is as follows: hosts: - "192.168.1.2" - "192.168.1.3" - "192.168.1.4"
	port	2181
execution_procedure	-	This parameter is left blank by default, indicating that the script is executed once to synchronize service data. Value options are **firststep** and **secondstep**. ● **firststep**: Only the temporary replication table is created. The auxiliary ZooKeeper can synchronize data from the original cluster to the temporary table in real time. ● **secondstep**: data in the temporary replication table is attached to the local table of the destination cluster. **CAUTION** If this parameter is set to **secondstep**, O&M personnel and users need to confirm that ClickHouse-related services have been stopped before script execution.
onereplica_use_auxiliaryzookeeper	-	● If this parameter is set to **1**, temporary tables are created for only one replica of each shard. ● If this parameter is set to **0**, temporary tables are created for two replicas of each shard.
migrate_timeout	-	ClickHouse data migration timeout. If the migration duration exceeds the specified value, the migration is considered complete and the temporary table clearing starts. The default value is **1440**, in minutes. Set this parameter based on the data volume. **NOTE** This parameter applies to MRS 3.3.0-LTS.1.1.

3. Stop the ClickHouse service of the source cluster.

4. Run the following command to migrate data and wait until the script execution is complete:

   ./clickhouse_migrate_data.sh -f yaml_file

   Enter the usernames and passwords of the source and destination clusters.

5. After the script is executed successfully, perform the following steps to check whether the migrated data in the source cluster is consistent with that in the destination cluster based on the migration result logs:

   Log in to the ClickHouse client node in the destination cluster and go to the *Client installation directory*/**ClickHouse/clickhouse_migration_tool/clickhouse-data-migration/comparison_result** directory.

   Compare the following result file information to check the data consistency between the source cluster and the destination cluster:

   - **source_cluster_table_info**: statistics of data migrated from the source cluster
   - **destination_cluster_table_info**: statistics of data migrated to the destination cluster
   - **compare_result_file.txt**: data consistency comparison result before and after migration

   If the data is inconsistent before and after the migration, clear the data in the table of the destination cluster and migrate the data in the table separately or manually.

   In addition, you can log in to the ClickHouse databases of the source and destination clusters to manually check whether the number of table data records and partitions are consistent.

6. Log in to FusionInsight Manager of the destination cluster and delete the ZooKeeper information added to **clickhouse-config-customize** in 2.

   Click **Save**. In the displayed dialog box, click **OK**.

7. After data migration is complete, switch services to the target ClickHouse cluster.

8. Go to *Client installation directory*/**ClickHouse/clickhouse_migration_tool/clickhouse-data-migration** and *Client installation directory*/**ClickHouse/clickhouse_migration_tool/clickhouse-metadata-migration** on the ClickHouse node in the destination cluster.

   vi example_config.yaml

   Delete the password from the configuration file to prevent password leakage.

NOTE

If service data migration fails, perform the following steps:

1. Locate the failure cause. Specifically, check whether any parameters in the configuration file are incorrectly configured.
   - If yes, reconfigure the parameters and perform service data migration.
   - If no, go to 2.
2. Run the **drop table** *table_name* command to delete the data tables related to the table from the node that fails to be migrated in the destination cluster.
3. Run the **show create table** *table_name* command to query the table creation statements related to the table in the source cluster and create the table in the destination cluster again.
4. Set the names of the tables that fail to be migrated in the service data migration configuration file based on the **databases** and **tables** parameters in Table 3-24 and run the service data migration command again. If the command fails to execute, contact O&M personnel.

3.7.6 Expanding the Disk Capacity of the ClickHouse Node

With the service volume increase, the data disk capacity of the ClickHouse node cannot meet service requirements and needs to be expanded.

CAUTION

For a pay-per-use MRS cluster, the billing mode cannot be changed to yearly/monthly after the disk capacity is expanded.

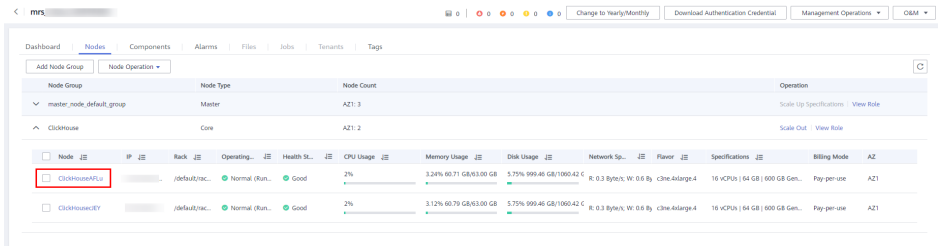
Prerequisites

- The ClickHouse cluster and instances are normal.
- You have evaluated the data disk capacity of the ClickHouse node to be expanded.

Expanding the Capacity of a Data Disk

Step 1 Log in to the MRS console. In the left navigation pane, choose **Active Clusters** and click a cluster name.

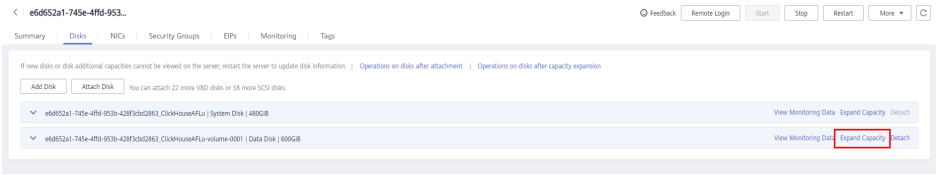
Step 2 Click **Nodes**. In the corresponding ClickHouse node group, click the name of the node to be expanded. The **Disks** page is displayed.



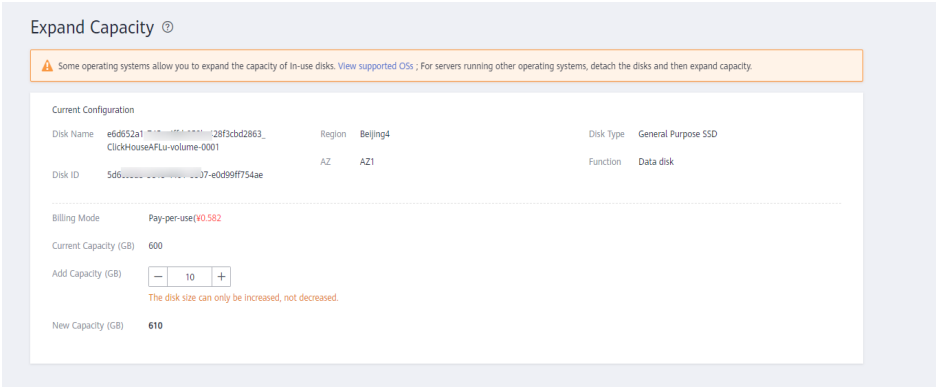
Step 3 In the row of the target data disk, click **Expand Capacity**.

NOTE

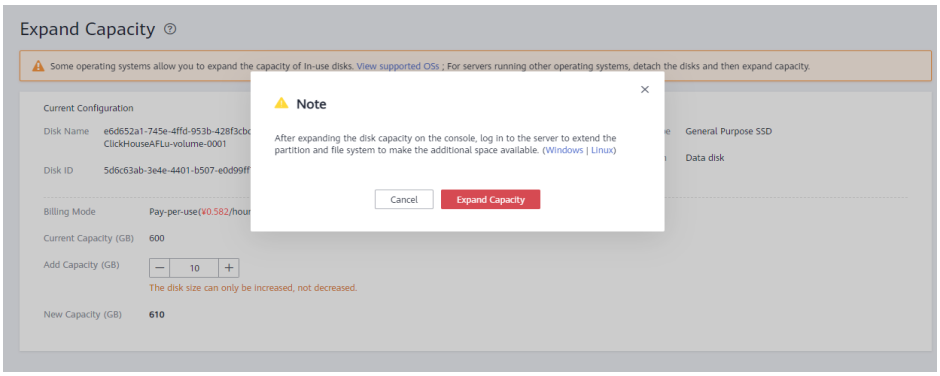
If only the system disk is displayed on the current page and no data disk exists, the data disk of the ClickHouse node cannot be expanded using this method.

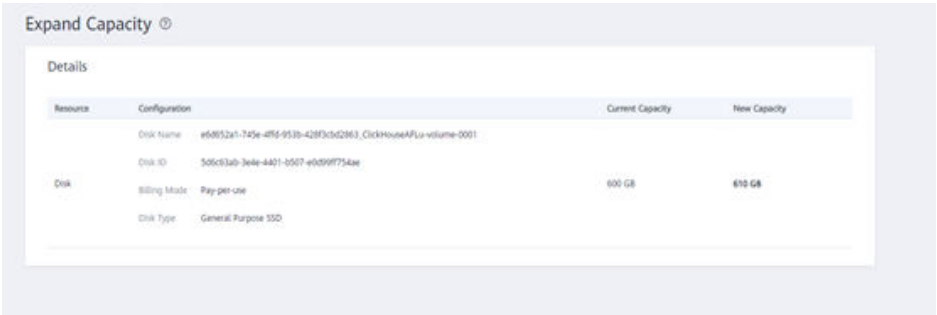


Step 4 Modify the disk capacity to be added in **Add Capacity (GB)** and click **Next**.



Step 5 Read the note for capacity expansion carefully, click **Expand Capacity**, confirm the information about the expanded disk capacity, and click **Submit**.





Step 6 Log in to the expanded ClickHouse node as user **root** and run the **df -hl** command to view information about the existing data directory and disk partition.


```
[root@ClickHouseAFLu ~]# df -hl
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1       217G   38G  170G  19% /
devtmpfs         32G     0   32G   0% /dev
tmpfs            32G     0   32G   0% /dev/shm
tmpfs            32G   73M   32G   1% /run
tmpfs            32G     0   32G   0% /sys/fs/cgroup
/dev/vda5       9.8G   37M  9.3G   1% /tmp
/dev/vda7        59G  147M   56G   1% /srv/BigData
/dev/vda6       9.8G  583M  8.7G   7% /var
/dev/vda8       177G  154M  168G   1% /var/log
/dev/vdb1       590G   75M  590G   1% /srv/BigData/data1
tmpfs           6.3G     0  6.3G   0% /run/user/2000
```

The default format of a ClickHouse data directory is **/srv/BigData/data*N***. The preceding figure shows that the ClickHouse data directory is **/srv/BigData/data1** and the corresponding partition is **/dev/vdb1**.

Step 7 Perform the following operations to make the new disk capacity take effect.

- To add a partition, go to **Step 8**. Adding a partition is to allocate the added disk capacity to a new partition and mount a new ClickHouse data directory to the new partition. This operation does not interrupt services.
- To extend an existing partition, go to **Step 15**. Extending an existing partition is to allocate the added disk capacity to an existing partition. Services will be interrupted during the operation. You are advised to stop services before the operation.

Step 8 For details about how to add a partition, see "Creating a New MBR Partition" or "Creating a New GPT Partition" in **Extending Partitions and File Systems for Data Disks (Linux)**.

Step 9 Log in to the expanded ClickHouse node as user **root** and run the following commands to create a ClickHouse data directory and create a mount point for the new partition. It is recommended that the directories be numbered in ascending order based on the current number.

For example, if the current data directory is **/srv/BigData/data1**, the added directory should be **/srv/BigData/data2**.

cd /srv/BigData/

mkdir data2

cd data2

mkdir clickhouse

cd /srv/BigData/

chmod 750 -R data2

chown omm:wheel -R data2

Step 10 Run the following command to mount the new partition:

mount *Disk partition Mounted directory*

For example, if the new partition is **/dev/vdb2** and the mounted directory is **/srv/BigData/data2**, run the following command:

mount /dev/vdb2 /srv/BigData/data2

NOTE

If the ECS is restarted, the mounting will become invalid. You can set automatic partition mounting upon system startup by modifying the **/etc/fstab** file. For details, see **Configuring Automatic Mounting at System Start**.

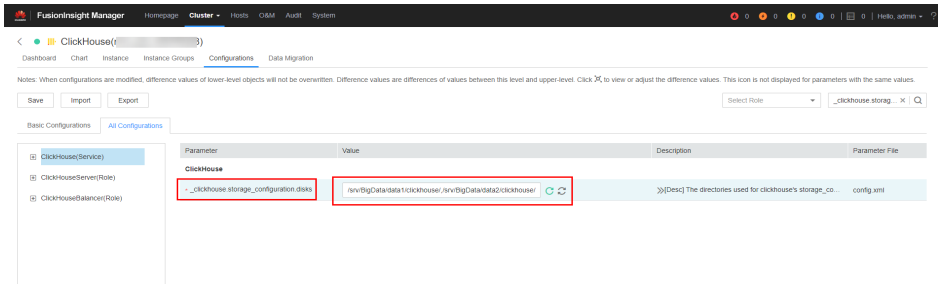
Step 11 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **ClickHouse** > **Configurations** > **All Configurations**.

Step 12 Search for **_clickhouse.storage_configuration.disks** and add the new ClickHouse data directory to the configuration item.

NOTE

Separate multiple directories with commas (,) and ensure that each directory ends with a slash (/).

For example, **/srv/BigData/data2/clickhouse/** is added in addition to the **/srv/BigData/data1/clickhouse/** directory. After the addition, the value is **/srv/BigData/data1/clickhouse/,/srv/BigData/data2/clickhouse/**.



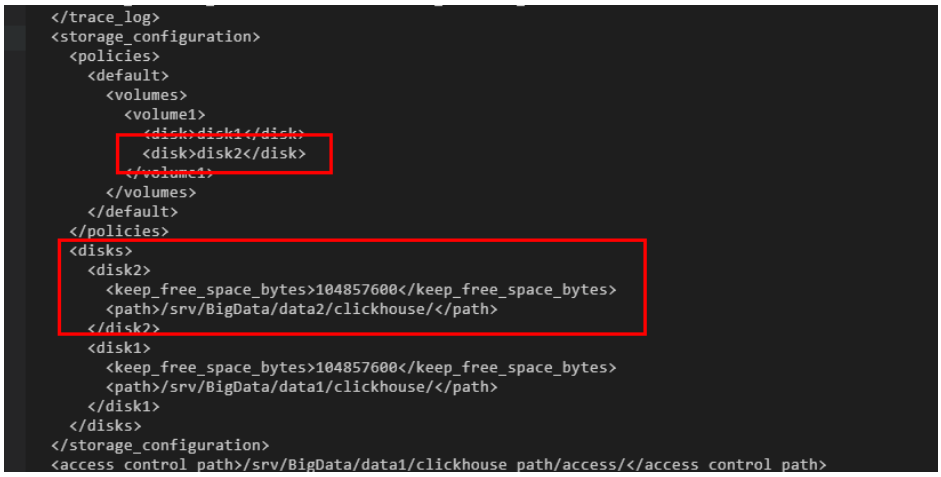
Step 13 After the new directory is added, click **Save** to save the configuration. Click **Dashboard**, choose **More** > **Synchronize Configuration**, and click **OK**.

Step 14 Log in to the expanded ClickHouse node, go to the following directory, and check whether the new data directory has been updated to the configuration file. After confirming that the information is correct, the operation is complete.

cd ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/*x_x_ClickHouse instance name***/etc**

cat config.xml

The following figure shows an example that the **/srv/BigData/data2/clickhouse/** directory has been added to the **config.xml** file.



Step 15 To extend an existing partition, ensure that the ClickHouse service has been stopped before the operation. Otherwise, services will be interrupted during the operation.

Step 16 Determine the partition to be extended based on **Step 6** and extend the partition by referring to "Extending an Existing MBR or GPT Partition" in **Extending Partitions and File Systems for Data Disks (Linux)**.

Step 17 After the existing partition is extended, run the ClickHouse service again.

----End

3.7.7 Backing Up and Restoring ClickHouse Data Using a Data File

Scenario

This section describes how to back up data by exporting ClickHouse data to a CSV file and restore data using the CSV file.

Prerequisites

- You have installed the ClickHouse client.
- You have created a user with related permissions on ClickHouse tables on Manager.
- You have prepared a server for backup.

Backing Up Data

Step 1 Log in to the node where the client is installed as the client installation user.

Step 2 Run the following command to go to the client installation directory:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the current user. The current user must have the permission to create ClickHouse tables. If Kerberos authentication is disabled for the current cluster, skip this step.

1. Run the following command if it is an MRS 3.1.0 cluster:

   export CLICKHOUSE_SECURITY_ENABLED=true

2. **kinit** *Component service user*

   Example: **kinit clickhouseuser**

Step 5 Run the ClickHouse client command to export the ClickHouse table data to be backed up to a specified directory.

clickhouse client --host *Host name/Instance IP address* **--secure --port 9440 --query="***Table query statement***" > *Path of the exported CSV file***

The following shows an example of backing up data in the **test** table to the **default_test.csv** file on the ClickHouse instance **10.244.225.167**.

clickhouse client --host 10.244.225.167 --secure --port 9440 --query="select * from default.test FORMAT CSV" > /opt/clickhouse/default_test.csv

Step 6 Upload the exported CSV file to the backup server.

----End

Restoring Data

Step 1 Upload the backup data file on the backup server to the directory where the ClickHouse client is located.

For example, upload the **default_test.csv** backup file to the **/opt/clickhouse** directory.

Step 2 Log in to the node where the client is installed as the client installation user.

Step 3 Run the following command to go to the client installation directory:

cd /opt/client

Step 4 Run the following command to configure environment variables:

source bigdata_env

Step 5 If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the current user. The current user must have the permission to create ClickHouse tables. If Kerberos authentication is disabled for the current cluster, skip this step.

1. Run the following command if it is an MRS 3.1.0 cluster:

   export CLICKHOUSE_SECURITY_ENABLED=true

2. **kinit** *Component service user*

   Example: **kinit clickhouseuser**

Step 6 Run the ClickHouse client command to log in to the ClickHouse cluster.

clickhouse client --host *Host name/Instance IP address* **--secure --port 9440**

Step 7 Create a table with the format corresponding to the CSV file.

CREATE TABLE [IF NOT EXISTS] *[database_name.]table_name* **[ON CLUSTER** *Cluster name***]**

(

name1 **[***type1***] [DEFAULT|***materialized***|ALIAS** *expr1***],**

name2 **[***type2***] [DEFAULT|***materialized***|ALIAS** *expr2***],**

...

) *ENGINE = engine*

Step 8 Import the content in the backup file to the table created in **Step 7** to restore data.

clickhouse client --host *Host name/Instance IP address* **--secure --port 9440 --query="insert into** *Table name* **FORMAT CSV"** < *CSV file path*

The following shows an example of restoring data from the **default_test.csv** backup file to the **test_cpy** table on the ClickHouse instance **10.244.225.167**.

clickhouse client --host 10.244.225.167 --secure --port 9440 --query="insert into default.test_cpy FORMAT CSV" < /opt/clickhouse/default_test.csv

----End

3.7.8 Configuring the Default ClickHouse User Password (MRS 3.1.2-LTS)

After a ClickHouse cluster is created, you can use the ClickHouse client to connect to the ClickHouse server. The default username is **default**.

This section describes how to set ClickHouse username and password after a ClickHouse cluster is successfully created.

NOTE

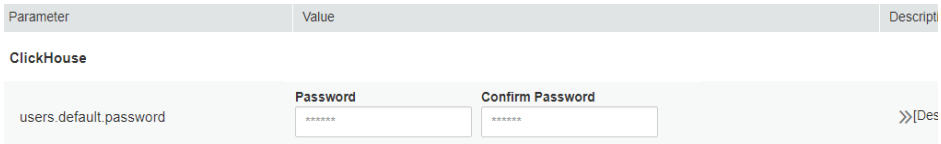
- This section applies to MRS 3.1.2.
- The **default** user is the ClickHouse administrator user that can be used only in normal mode (kerberos authentication is disabled).

Configuring the Password of the Default Account of a ClickHouse Cluster

Step 1 Log in to Manager and choose **Cluster** > **Services** > **ClickHouse**. Click the **Configurations** tab and then **All Configurations**.

Step 2 Search for the **users.default.password** parameter in the search box and change its password, as shown in **Figure 3-16**.

Figure 3-16 Changing the default user password



Step 3 Log in to the node where the client is installed and run the following command to switch to the client installation directory.

cd*Cluster client installation directory*

Step 4 Run the following command to configure environment variables:

source bigdata_env

Step 5 Log in to ClickHouse using the new password.

clickhouse client --host *IP address of the ClickHouse instance***--user** *default* **--password**

Enter the user password.

NOTE

To obtain the IP address of the ClickHouse instance, choose **Components** > **ClickHouse** > **Instances** on the cluster details page.

----End

3.7.9 Configuring the Default ClickHouse User Password (MRS 3.3.0-LTS or later)

After a ClickHouse cluster is created, you can use the ClickHouse client to connect to the ClickHouse server.

Set the passwords of default ClickHouse users **default** and **clickhouse** after creating a ClickHouse cluster in normal mode.

NOTE

- This section applies to MRS 3.3.0-LTS or later.
- **default** and **clickhouse** are default internal administrators of a ClickHouse cluster in normal mode (with Kerberos authentication disabled).
- For a ClickHouse cluster in normal mode, if the default passwords of the default users **default** and **clickhouse** have been changed and the ClickHouseServer node is reinstalled, the passwords will be reset. You need to change the passwords again.

Changing the Password of the Default ClickHouse User

Step 1 Log in to the node where ClickHouse is installed as user **root**, switch to user **omm**, and go to the **$BIGDATA_HOME/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/clickhouse_change_password** directory.

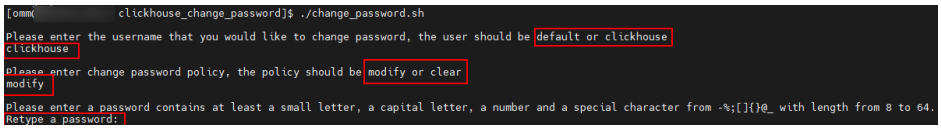
su - omm

cd $BIGDATA_HOME/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/clickhouse_change_password

Step 2 Run the following command to change the password of user **default** or **clickhouse**:

./change_password.sh

In the following figure, user **clickhouse** is used as an example. Enter **clickhouse** and its password as prompted, and wait until the password is changed.



NOTE

The password must meet the following complexity requirements:

- Contains 8 to 64 characters.
- Contains at least one lowercase letter, one uppercase letter, one number, and one special character (-%;[]{}@_).

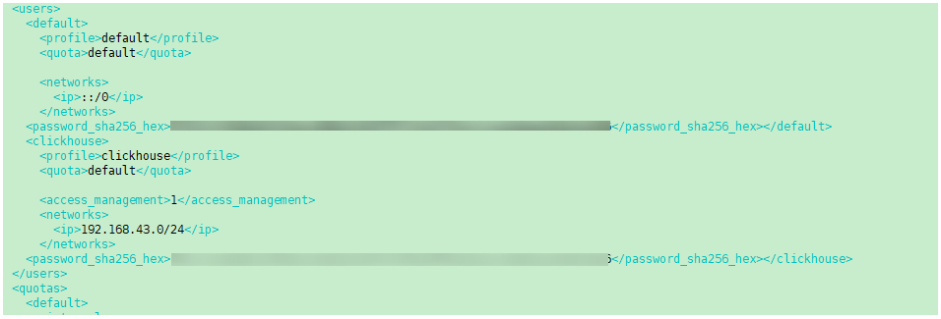
Step 3 Check the changed password.

- For MRS 3.3.0-LTS version: Log in to the ClickHouse Server node and check the value of **password_sha256_hex** in the **${BIGDATA_HOME}/FusionInsight_ClickHouse_*/*_ClickHouseServer/etc/users.xml** file. The value is the new password.

  cd ${BIGDATA_HOME}/*FusionInsight_ClickHouse_*/*_ClickHouseServer***/etc/**

  vi users.xml

  As shown in the following figure, the new password is stored in the **password_sha256_hex** file.

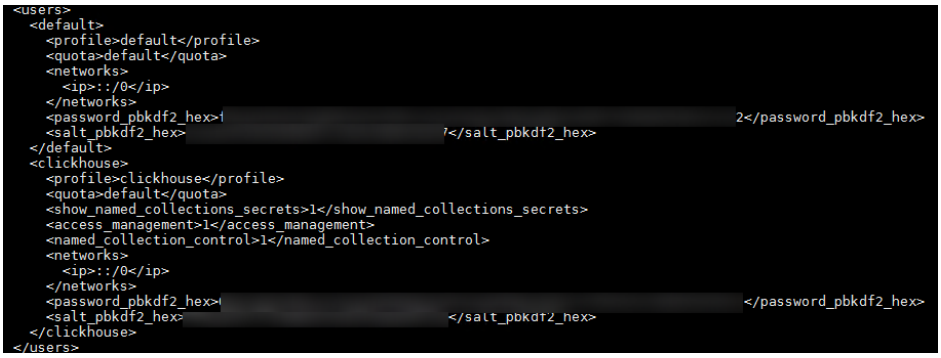


- For MRS 3.3.1-LTS and later versions: Log in to the ClickHouse Server node and check the value of **password_PBKDF2_hex** in the **${BIGDATA_HOME}/FusionInsight_ClickHouse_*/*_ClickHouseServer/etc/users.xml** file. The value is the new password.

  cd ${BIGDATA_HOME}/*FusionInsight_ClickHouse_*/*_ClickHouseServer***/etc/**

  vi users.xml

  As shown in the following figure, the new password is stored in the **password_PBKDF2_hex** file.



----End

3.7.10 Clearing the Passwords of Default ClickHouse Users

After a ClickHouse cluster in normal mode is created, clear the passwords of default users **default** and **clickhouse**.

NOTE

- This topic is available in MRS 3.3.0 or later only.
- **default** and **clickhouse** are default internal administrators of a ClickHouse cluster in normal mode (with Kerberos authentication disabled).

Clearing Default User Passwords

Step 1 Log in to FusionInsight Manager, choose **Cluster** > **Service** > **ClickHouse** > **Configurations** > **All Configurations**, search for **ALLOW_CLEAR_INTERNAL_ACCOUNT_PASSWORD**, and change the value to **true**.

Step 2 Log in to the node where ClickHouse is installed as user **root**, switch to user **omm**, and go to the **$BIGDATA_HOME/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/clickhouse_change_password** directory.

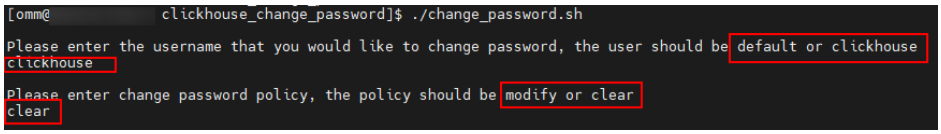
su - omm

cd $BIGDATA_HOME/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/clickhouse_change_password

Step 3 Run the following command to clear the password of user **default** or **clickhouse**:

./change_password.sh

In the following figure, user **clickhouse** is used as an example. Enter **clickhouse** and its password as prompted, and wait until the password is cleared.



Step 4 Verify that the password is cleared.

Log in to the ClickHouse Server node and check whether the value of **password** in the **${BIGDATA_HOME}/FusionInsight_ClickHouse_*/*_ClickHouseServer/etc/users.xml** file is empty.

cd ${BIGDATA_HOME}/*FusionInsight_ClickHouse_*/*_ClickHouseServer***/etc/**

vi users.xml

The following is an example.


```
<clickhouse>
  <profile>clickhouse</profile>
  <quota>default</quota>
  <password/>
  <access_management>1</access_management>
  <networks>
    <ip>192.168.67.0/24</ip>
  </networks>
</clickhouse>
```

----End

3.8 Common ClickHouse SQL Syntax

3.8.1 CREATE DATABASE: Creating a Database

This section describes the basic syntax and usage of the SQL statement for creating a ClickHouse database.

Basic Syntax

CREATE DATABASE [IF NOT EXISTS] *database_name* **[ON CLUSTER** *ClickHouse cluster name***]**

NOTE

The syntax **ON CLUSTER** *ClickHouse cluster name* enables the Data Definition Language (DDL) statement to be executed on all instances in the cluster at a time. You can run the following statement to obtain the cluster name from the **cluster** field:

select cluster,shard_num,replica_num,host_name from system.clusters;

Example

```
-- Create a database named test.
CREATE DATABASE test ON CLUSTER default_cluster;
-- After the creation is successful, run the query command for verification.
show databases;
┌─name────┐
│ default │
│ system  │
│ test    │
└─────────┘
```

3.8.2 CREATE TABLE: Creating a Table

This section describes the basic syntax and usage of the SQL statement for creating a ClickHouse table.

Basic Syntax

- Method 1: Creating a table named **table_name** in the specified **database_name** database.

  If the table creation statement does not contain **database_name**, the name of the database selected during client login is used by default.

CREATE TABLE [IF NOT EXISTS] *[database_name.]table_name* **[ON CLUSTER** *ClickHouse cluster name***]**

(

name1 **[***type1***] [DEFAULT|MATERIALIZED|ALIAS** *expr1***],**

name2 **[***type2***] [DEFAULT|MATERIALIZED|ALIAS** *expr2***],**

...

) *ENGINE = engine_name()*

[**PARTITION BY** *expr_list*]

[**ORDER BY** *expr_list*]

CAUTION

You are advised to use **PARTITION BY** to create table partitions when creating a ClickHouse table. The ClickHouse data migration tool works on table partitions. If you do not use **PARTITION BY** to create table partitions in table creation, the table data cannot be migrated on the UI in **Migrating Data Between ClickHouseServer Nodes in a Cluster**.

- Method 2: Creating a table with the same structure as **database_name2.table_name2** and specifying a different table engine for the table

  If no table engine is specified, the created table uses the same table engine as **database_name2.table_name2**.

  CREATE TABLE [IF NOT EXISTS] *[database_name.]table_name* **AS** [*database_name2.*]*table_name2* [ENGINE = *engine_name*]

- Method 3: Using the specified engine to create a table with the same structure as the result of the **SELECT** clause and filling it with the result of the **SELECT** clause

  CREATE TABLE [IF NOT EXISTS] *[database_name.]table_name ENGINE = engine_name* **AS SELECT** ...

Example

```
-- Create a table named test in the default database and default_cluster cluster.
CREATE TABLE default.test ON CLUSTER default_cluster
(
    `EventDate` DateTime,
    `id` UInt64
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/test', '{replica}')
PARTITION BY toYYYYMM(EventDate)
ORDER BY id
```

3.8.3 INSERT INTO: Inserting Data into a Table

This section describes the basic syntax and usage of the SQL statement for inserting data to a table in ClickHouse.

Basic Syntax

- Method 1: Inserting data in standard format

INSERT INTO *[database_name.]table* [(*c1, c2, c3*)] **VALUES** (*v11, v12, v13*), (*v21, v22, v23*), ...

- Method 2: Using the **SELECT** result to insert data

  INSERT INTO *[database_name.]table* [(c1, c2, c3)] **SELECT** ...

Example

```
-- Insert data into the test2 table.
insert into test2 (id, name) values (1, 'abc'), (2, 'bbbb');
-- Query data in the test2 table.
select * from test2;
┌─id─┬─name─┐
│  1 │ abc  │
│  2 │ bbbb │
└────┴──────┘
```

3.8.4 DELETE: Lightweight Deleting Table Data

This topic describes the basic syntax and usage of the SQL statement for deleting table data in a lightweight way.

NOTE

This topic is available for MRS 3.3.0 or later only.

Basic Syntax

DELETE FROM *[db.]table [ON CLUSTER cluster]* **WHERE** *expr*

Example

- Create a table.

```
CREATE TABLE default.test_ligtwight_delete
(
`id`    Int32,
`pdate` Date,
`name`  String,
`class` Int32
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/distributed_tests/{shard}/test_ligtwight_delete',
'{replica}')
PARTITION BY toYYYYMM(pdate)
PRIMARY KEY id
ORDER BY id
SETTINGS index_granularity = 8192, vertical_merge_algorithm_min_rows_to_activate = 1,
vertical_merge_algorithm_min_columns_to_activate = 1, min_rows_for_wide_part = 1,
min_bytes_for_wide_part = 1;
```

- Insert data.

```
insert into default.test_ligtwight_delete select rand(), rand() % 365, rand(), rand() from numbers(10);
```

- Delete data.

```
delete from default.test_ligtwight_delete where id > 0;
```

Precautions

- Deleted rows are immediately marked as deleted and automatically filtered out from all subsequent queries. Data cleanup occurs asynchronously in the background. This function is only available for the MergeTree table engine series.

- Currently, only lightweight deletion is supported for local tables and replication tables, but not for distributed tables.
- The lightweight deletion performance depends on the number of merge and mutation (alter table update/delete) tasks. Mutation tasks in a queue have the lowest priority (mutation tasks in the same table are executed serially). The number of concurrent delete tasks is directly affected by the execution of the merge tasks.
- The number of parts in the table also determines the lightweight deletion performance. The more parts, the slower the deletion.
- Data parts in Wide format can be deleted quickly, and those in compact files can be deleted slowly because all column data is stored in one file.

3.8.5 SELECT: Querying Table Data

This section describes the basic syntax and usage of the SQL statement for querying table data in ClickHouse.

Basic Syntax

SELECT [**DISTINCT**] expr_list

[**FROM** [*database_name*.]*table* | (subquery) | table_function] [**FINAL**]

[SAMPLE sample_coeff]

[ARRAY **JOIN** ...]

[**GLOBAL**] [**ANY**|**ALL**|**ASOF**] [**INNER**|**LEFT**|**RIGHT**|**FULL**|**CROSS**] [**OUTER**|SEMI|ANTI] **JOIN** (subquery)|**table** (**ON** <expr_list>)|(**USING** <column_list>)

[PREWHERE expr]

[**WHERE** expr]

[**GROUP BY** expr_list] [**WITH** TOTALS]

[**HAVING** expr]

[**ORDER BY** expr_list] [**WITH** FILL] [**FROM** expr] [**TO** expr] [STEP expr]

[**LIMIT** [offset_value,]n **BY** columns]

[**LIMIT** [n,]m] [**WITH** TIES]

[**UNION ALL** ...]

[**INTO** OUTFILE filename]

[FORMAT format]

Example

```
-- View ClickHouse cluster information.
select * from system.clusters;
-- View the macros set for the current node.
select * from system.macros;
-- Check the database capacity.
select
sum(rows) as "Total number of rows",
formatReadableSize(sum(data_uncompressed_bytes)) as "Original size",
```



```
formatReadableSize(sum(data_compressed_bytes)) as "Compression size",
round(sum(data_compressed_bytes) / sum(data_uncompressed_bytes) * 100,
0) "Compression rate"
from system.parts;
-- Query the capacity of the test table. Add or modify the where clause based on the site requirements.
select
sum(rows) as "Total number of rows",
formatReadableSize(sum(data_uncompressed_bytes)) as "Original size",
formatReadableSize(sum(data_compressed_bytes)) as "Compression size",
round(sum(data_compressed_bytes) / sum(data_uncompressed_bytes) * 100,
0) "Compression rate"
from system.parts
where table in ('test')
and partition like '2020-11-%'
group by table;
```

3.8.6 ALTER TABLE: Modifying a Table Structure

This section describes the basic syntax and usage of the SQL statement for modifying a table structure in ClickHouse.

Basic Syntax

ALTER TABLE [*database_name*].*name* [**ON CLUSTER** *cluster*] **ADD**|**DROP**|**CLEAR**|**COMMENT**|**MODIFY COLUMN** ...

NOTE

ALTER supports only *MergeTree, Merge, and Distributed engine tables.

Example

```
-- Add the test01 column to the t1 table.
ALTER TABLE t1 ADD COLUMN test01 String DEFAULT 'defaultvalue';
-- Query the modified table t1.
desc t1
┌─name─────┬─type───┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ id       │ UInt8  │              │                    │         │                  │                │
│ name     │ String │              │                    │         │                  │                │
│ address  │ String │              │                    │         │                  │                │
│ test01   │ String │ DEFAULT      │ 'defaultvalue'     │         │                  │                │
└──────────┴────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
-- Change the type of the name column in the t1 table to UInt8.
ALTER TABLE t1 MODIFY COLUMN name UInt8;
-- Query the modified table t1.
desc t1
┌─name─────┬─type───┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ id       │ UInt8  │              │                    │         │                  │                │
│ name     │ UInt8  │              │                    │         │                  │                │
│ address  │ String │              │                    │         │                  │                │
│ test01   │ String │ DEFAULT      │ 'defaultvalue'     │         │                  │                │
└──────────┴────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
-- Delete the test01 column from the t1 table.
ALTER TABLE t1 DROP COLUMN test01;
-- Query the modified table t1.
desc t1
┌─name─────┬─type───┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ id       │ UInt8  │              │                    │         │                  │                │
│ name     │ UInt8  │              │                    │         │                  │                │
│ address  │ String │              │                    │         │                  │                │
└──────────┴────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
```


3.8.7 ALTER TABLE: Modifying Table Data

- Exercise caution when doing delete, update, and mutation operations.

  The update and delete of standard SQL statements are synchronous operations. That is, the client needs to wait for the server to return the execution results (usually an **int** value). In contrast, the update and delete of ClickHouse are asynchronous operations. When an update statement is processed, the server immediately returns the request status: success or fail, while the operation is not complete. At that time, the update request is accepted and queued in the background. As a result, the operation may be overwritten, and atomicity of operations cannot be ensured.

  To solve this problem, you are advised to use the ReplacingMergeTree, CollapsingMergeTree, and VersionedCollapsingMergeTree engines to update and delete data. For details, see **https://clickhouse.com/docs/en/engines/table-engines/mergetree-family/collapsingmergetree**.

- Try to avoid adding or deleting data columns.

  Plan the number of columns for future use, reserve enough columns to avoid a large number of alter table modify operations during service running in the production system. Otherwise, unpredictable performance problem and data inconsistency may occur.

3.8.8 DESC: Querying a Table Structure

This section describes the basic syntax and usage of the SQL statement for querying a table structure in ClickHouse.

Basic Syntax

DESC|**DESCRIBE TABLE** [*database_name*.]*table* [**INTO** OUTFILE filename] [FORMAT format]

Example

Query the **t1** table structure:

```
desc t1;
┌─name────┬─type───┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ id      │ UInt8  │              │                    │         │                  │                │
│ name    │ UInt8  │              │                    │         │                  │                │
│ address │ String │              │                    │         │                  │                │
└─────────┴────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
```

3.8.9 DROP: Deleting a Table

This section describes the basic syntax and usage of the SQL statement for deleting a ClickHouse table.

Basic Syntax

DROP [**TEMPORARY**] **TABLE** [**IF EXISTS**] [*database_name*.]*name* [**ON CLUSTER** *cluster*] [SYNC]

Example

```
-- Delete the t1 table.
drop table t1 SYNC;
```

NOTE

When you delete a replication table, create a path on ZooKeeper to store related data. The default library engine of ClickHouse is the atomic database engine. After a table in the atomic database is deleted, it is not deleted immediately but deleted 480 seconds later. To resolve this issue, when deleting a table, add the **SYNC** field to the deletion command, for example, **drop table** *t1* **SYNC;**.

This issue does not occur when a local or distributed table is deleted. The **SYNC** field is not required in your deletion command, for example, **drop table** *t1***;**.

3.8.10 SHOW: Displaying Information About Databases and Tables

This section describes the basic syntax and usage of the SQL statement for displaying information about databases and tables in ClickHouse.

Basic Syntax

show databases

show tables

Example

```
-- Query database information.
show databases;
┌─name───┐
│ default │
│ system  │
│ test    │
└─────────┘
-- Query table information.
show tables;
┌─name─┐
│ t1   │
│ test │
│ test2 │
│ test5 │
└──────┘
```

3.8.11 UPSERT: Writing Data

This topic describes the basic SQL syntax and usage of the upsert function when ClickHouse data is written.

NOTE

This topic is available for MRS 3.3.0 or later only.

Basic Syntax

- INSERT VALUES

  UPSERT INTO *[database_name.]table* [(*c1, c2, c3*)] **VALUES** (*v11, v12, v13*), (*v21, v22, v23*), ...

- INSERT SELECT

  UPSERT INTO *[database_name.]table* [(c1, c2, c3)] **SELECT** ...

Example

- Create a table.

```
CREATE TABLE default.upsert_tab ON CLUSTER default_cluster
(
`id`    Int32,
`pdate`  Date,
`name`   String
)ENGINE = ReplicatedMergeTree('/clickhouse/tables/default/{shard}/upsert_tab', '{replica}')
PARTITION BY toYYYYMM(pdate)
PRIMARY KEY id
ORDER BY id
SETTINGS index_granularity = 8192;
```

- Upsert data.

```
Upsert into upsert_tab(id, pdate, name) values (1, rand() % 365, 'abc'), (2, rand() % 365, 'bcd'), (1, rand() % 365, 'def');
```

- Query data in the **test_upsert** table.

```
select * from upsert_tab;
┌─id─┬──────pdate─┬─name─┐
│  2 │ 1970-06-09 │ bcd  │
│  1 │ 1970-11-30 │ def  │
└────┴────────────┴──────┘
```

- Upsert for transactions

  Similar to other SQL syntax, Upsert also supports explicit and implicit transactions. Before using transactions, you need to enable the transaction function.

Precautions

- When creating MergeTree and ReplicatedMergeTree tables, specify the primary key or order by field as the unique key for deduplication. If no primary key is specified and only the order by property is specified during table creation, the order by field is used for deduplication.
- The key for deduplication must be sharded in advance to ensure that same key fields are in the same shard to ensure deduplication accuracy.

3.9 Common Issues About ClickHouse

3.9.1 How Do I Do If the Disk Status Displayed in the System.disks Table Is fault or abnormal?

Question

How do I do if the disk status displayed in the System.disks table is fault or abnormal?

Answer

This problem is caused by I/O errors on the disk. To rectify the fault, perform the following steps:

- Method 1: Log in to FusionInsight Manager and check whether an alarm is generated indicating that the disk I/O is abnormal. If yes, replace the faulty disk by referring to the alarm help.
- Method 2: Log in to FusionInsight Manager and restart the ClickHouse instance to restore the disk status.

  NOTE

  If an I/O error occurs but the disk is not replaced, the disk status will still turn to fault or abnormal.

3.9.2 How Do I Migrate Data from Hive/HDFS to ClickHouse?

Question

How do I migrate Hive/HDFS data to ClickHouse?

Answer

You can export data from Hive as CSV files and import the CSV files to ClickHouse.

1. Export data from Hive as CSV files.

   hive -e "select * from db_hive.student limit 1000"| tr "\t" "," > /data/bigdata/hive/student.csv;

2. Import the CSV files to the **student_hive** table in the default database of ClickHouse.

   clickhouse --client --port 9002 --password *xxx* **-m --query='INSERT INTO default.student_hive FORMAT CSV' < /data/bigdata/hive/student.csv**

   Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

3.9.3 How Do I Migrate Data from OBS/S3 to ClickHouse?

Question

How do I migrate data from OBS/S3 to ClickHouse?

Answer

1. Query data stored in S3.

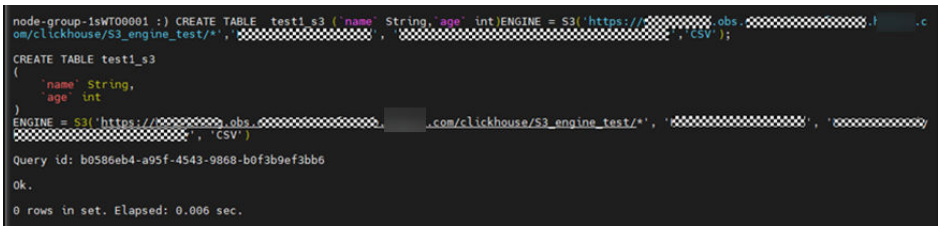
   select * from s3(*path [,access_key_id, secret_access_key] [,format] [,structure]*)

NOTE

- **path**: bucket URL with a file path.
- **format**: The file format.
- **access_key_id** and **secret_access_key**: long-term credentials of an account. Credentials can be used to authenticate requests. These parameters are optional. If no credential is specified, credentials are read from the configuration file.
- **structure**: The table schema.

2. Obtain data from S3 to create a table.

   CREATE TABLE test1_s3 (name String, value UInt32) ENGINE = S3(*path, [access_key_id, secret_access_key,] format*)



3. View the created table.

   select * from test1_s3

3.9.4 An Error Is Reported in Logs When the Auxiliary ZooKeeper or Replica Data Is Used to Synchronize Table Data

Question

An error is reported in logs when the auxiliary ZooKeeper or replica data is used to synchronize table data.

```
DB::Exception: Cannot parse input: expected 'quorum:' before: 'merge_type: 2'...
Too many parts (315). Merges are processing significantly slower than inserts...
```

Answer

The versions of replication table replicas are inconsistent, causing compatibility issues. The table schema contains TTL statements. TTL_DELETE is added in versions later than ClickHouse 20.9, which cannot be identified in earlier versions.

This issue occurs when the replication table replica of a later version is elected as the leader.

You can modify the **config.xml** file of ClickHouse of a later version to avoid such an issue. Ensure that the replication table replicas are the same as those of ClickHouse.

3.9.5 How Do I Grant the Select Permission at the Database Level to ClickHouse Users?

Procedure

Step 1 Log in to the node where the ClickHouse client is installed in the MRS cluster and run the following commands:

su - omm

source *{Client installation directory}*/**bigdata_env**

kinit *Component user* (You do not need to run the **kinit** command for normal clusters.)

clickhouse client --host *IP address of the ClickHouse node* **--port 9000 -m --user clickhouse –password '***Password of the ClickHouse user***'**

NOTE

- View the password of the ClickHouse user.

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Instance** and click any ClickHouseServer role name. Go to the **Dashboard** tab page of ClickHouseServer, click the **users.xml** file in **Configuration File** area, and view the password of the ClickHouse user.
- Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

Step 2 You can use either of the following methods to create a role with the read-only permission for a specified database:

Method 1

1. Creating a role with the read-only permission for a specified database (the **default** database is used as an example)

   create role ck_role on cluster default_cluster;

   GRANT SELECT ON default.* TO ck_role on cluster default_cluster;
2. Creating a common user

   CREATE USER user_01 on cluster default_cluster IDENTIFIED WITH PLAINTEXT_PASSWORD BY '*password***';**
3. Granting the read-only permission role to a common user

   GRANT ck_role to user_01 on cluster default_cluster;
4. Viewing user permissions

   show grants for user_01;

   select * from system.grants where role_name = 'ck_role';

Method 2

Creating a user with the read-only permission for a specified database

1. Creating a user:

   CREATE USER user_01 on cluster default_cluster IDENTIFIED WITH PLAINTEXT_PASSWORD BY '*password*';

2. Granting the query permission on a specified database to the created user:

   grant select on default.* to user_01 on cluster default_cluster;

3. Querying user permissions:

   select * from system.grants where user_name = 'user_01';

----End

3.9.6 How Do I Quickly Restore ClickHouse When Concurrent Requests Are Stacked for a Long Time?

Symptom

When there are too many concurrent ClickHouse requests, new requests cannot be handled, affecting service availability. The error information is as follows:

```
DB::Exception: Too many simultaneous queries. Maximum: 100
```

Procedure

Step 1 Log in to the ClickHouse client node and connect to the server where the error is reported. For details, see **ClickHouse Client Practices**.

Step 2 Run the following command to check the number of concurrent SQL jobs that are being executed:

select count(*) from system.processes;

Verify that the obtained value is greater than or equal to 100.

Step 3 Run the following command to stop large query SQL statements that are being executed:

kill query where query_kind='Select' and elapsed > 60;

Step 4 Wait for 30 seconds and run the following command to collect statistics on the SQL jobs that are being executed:

select count(*) from system.processes;

Check whether the obtained value is less than 60.

Step 5 If the number of concurrent jobs is greater than 60, log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse** > **Instances**, select the ClickHouseServer instance that reports the error, and choose **More** > **Restart Instance**.

----End

4 Using DBService

4.1 Configuring SSL for the HA Module

Scenario

This section describes how to manually configure SSL for the HA module of DBService in the cluster where DBService is installed.

NOTE

After this operation is performed, if you need to restore the SSL configuration, go to **Restoring SSL for the HA Module**.

Prerequisites

- The cluster has been installed.
- The **root-ca.crt** and **root-ca.pem** files in the **$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/security** directory on the active and standby DBService nodes are the same.

Procedure

Step 1 Log in to the DBService node where SSL needs to be configured as user **omm**.

Step 2 Go to the **$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/sbin/** directory and run the following command:

./proceed_ha_ssl_cert.sh *DBService installation directoryService IP address of the node*

Example:

cd $BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/sbin/

./proceed_ha_ssl_cert.sh $BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0 10.10.10.10

NOTE

$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0 is the installation directory of DBService. Modify it based on site requirements.

Step 3 Go to the **$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/ha/module/hacom/script/** directory and run the following command to restart the HA process:

./stop_ha.sh

./start_ha.sh

Step 4 Run the following command on the preceding node to obtain the PID of the HA process:

ps -ef |grep "ha.bin" |grep DBSERVICE

Step 5 Run the following command to check whether the protocol is changed to TCP:

netstat -nap | grep *pid* **| grep -v unix**

- If yes, no further action is required.
- If no, go to **Step 2**.

```
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 127.0.0.1:20054         0.0.0.0:*               LISTEN      11896/ha.bin
tcp        0      0 10.10.10.10:20052       10.10.10.14:20052       ESTABLISHED 11896/ha.bin
tcp        0      0 10.10.10.10:20053       10.10.10.14:20053       ESTABLISHED 11896/ha.bin
```

----End

4.2 Restoring SSL for the HA Module

Scenario

This section describes how to restore SSL for the HA module of DBService in the cluster where DBService is installed.

Prerequisites

SSL has been enabled for the HA module of DBService.

NOTE

Check whether SSL is enabled for the HA module of DBService.

Check **$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/ha/module/hacom/conf/hacom.xml**. If the file contains **<hadataprotocol value="ssl"></hadataprotocol>**, SSL is enabled.

Procedure

Step 1 Log in to the DBService node where SSL needs to be restored as user **omm**.

Step 2 Run the following commands to restore the DBService configuration file **hacom_local.xml**:

cd $BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/ha/local/hacom/conf/

cp hacom_local.xml $BIGDATA_HOME/tmp/

cat hacom_local.xml | grep "ssl>" -n | cut -d':' -f1 | xargs | sed 's/ /,/g' |xargs -n 1 -i sed -i '{}d' hacom_local.xml

Step 3 Run the following commands to restore the DBService configuration file **hacom.xml**:

cd $BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/ha/module/hacom/conf/

cp hacom.xml $BIGDATA_HOME/tmp/

sed -i 's#<hadataprotocol.*#<hadataprotocol value="udp"/>#g' hacom.xml

sed -i 's#<rpcsupportssl.*#<rpcsupportssl value="true"/>#g' hacom.xml

NOTE

$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0 is the installation directory of DBService. Modify it based on the upgrade environment.

Step 4 Go to the **$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/ha/module/hacom/script/** directory and run the following command to restart the HA process:

./stop_ha.sh

./start_ha.sh

Step 5 Run the following command to obtain the PID of the HA process:

ps -ef |grep "ha.bin" |grep DBSERVICE

Step 6 Run the following command to check whether the protocol is changed to TCP:

netstat -nap | grep *pid* | grep -v unix

- If yes, no further action is required.
- If no, contact O&M support.

```
[omm@host03]\>netstat -nap | grep 49989
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 127.0.0.1:20054         0.0.0.0:*               LISTEN      49989/ha.bin
udp        0      0 10.10.10.10:20052       0.0.0.0:*                           49989/ha.bin
udp        0      0 10.10.10.10:20053       0.0.0.0:*                           49989/ha.bin
```

----End

4.3 Configuring the Timeout Interval of DBService Backup Tasks

Scenario

The default timeout interval of DBService backup tasks is 2 hours. When the data volume in DBService is too large, the backup task may fail to be executed because the timeout interval is reached.

This section describes how to customize the timeout interval of a DBService backup task.

Prerequisites

- Clusters have been properly installed.
- DBService is running properly.

Procedure

Step 1 Log in to the active OMS node as user **omm** using PuTTY. In the **${CONTROLLER_HOME}/etc/om/controller.properties** configuration file, change the value of **controller.backup.conf.script.execute.timeout** to **10000000** (Set the timeout interval based on the data volume of DBService).

Step 2 Log in to the active OMS node as user **omm** using PuTTY and repeat step **Step 1**.

Step 3 Log in to the active OMS node as user **omm** using PuTTY, run the following command to query the ID of **BackupRecoveryPluginProcess** process, and stop the process.

jps|grep -i BackupRecoveryPluginProcess

kill -9 *Queried process ID*

Step 4 Log in to FusionInsight Manager and perform the DBService backup task again.

Step 5 Run the following command to check whether the **BackupRecoveryPluginProcess** process is started:

jps|grep -i BackupRecoveryPluginProcess

----End

4.4 DBService Log Overview

Log Description

Log path: The default storage path of DBService log files is **/var/log/Bigdata/dbservice**.

- GaussDB: **/var/log/Bigdata/dbservice/DB** (GaussDB run log directory), **/var/log/Bigdata/dbservice/scriptlog/gaussdbinstall.log**

(GaussDB installation log), and **/var/log/gaussdbuninstall.log** (GaussDB uninstallation log).

- HA: **/var/log/Bigdata/dbservice/ha/runlog** (HA run log directory) and **/var/log/Bigdata/dbservice/ha/scriptlog** (HA script log directory)
- DBServer: **/var/log/Bigdata/dbservice/healthCheck** (Directory of service and process health check logs)

  /var/log/Bigdata/dbservice/scriptlog (run log directory), **/var/log/Bigdata/audit/dbservice/** (audit log directory)

Log archive rule: The automatic DBService log compression function is enabled. By default, when the size of logs exceeds 1 MB, logs are automatically compressed into a log file named in the following format: *<Original log file name>-[No.]*.**gz**. A maximum of 20 latest compressed files are reserved.

NOTE

Log archive rules cannot be modified.

Table 4-1 DBService log list

Type	Log File Name	Description
DBServer run log	dbservice_serviceCheck.log	Run log file of the service check script
	dbservice_processCheck.log	Run log file of the process check script
	backup.log	Run logs of backup and restoration operations (The DBService backup and restoration operations need to be performed.)
	checkHaStatus.log	Log file of HA check records
	cleanupDBService.log	Uninstallation log file (You need to uninstall DBService logs.)
	componentUserManager.log	Log file that records the adding and deleting operations on the database by users (Services that depend on DBService need to be added.)
	install.log	Installation log file
	preStartDBService.log	Pre-startup log file

Type	Log File Name	Description
	start_dbserver.log	DBServer startup operation log file (DBService needs to be started.)
	stop_dbserver.log	DBServer stop operation log file (DBService needs to be stopped.)
	status_dbserver.log	Log file of the DBServer status check (You need to execute the **$DBSERVICE_HOME/sbin/status-dbserver.sh** script.)
	modifyPassword.log	Run log file of changing the DBService password script. (You need to execute the **$DBSERVICE_HOME/sbin/modifyDBPwd.sh** script.)
	modifyDBPwd_yyyy-mm-dd.log	Run log file that records the DBService password change tool (You need to execute the **$DBSERVICE_HOME/sbin/modifyDBPwd.sh** script.)
	dbserver_switchover.log	Log for DBServer to execute the active/standby switchover script (the active/standby switchover needs to be performed)
GaussDB run log	gaussdb.log	Log file that records database running information
	gs_ctl-current.log	Log file that records operations performed by using the **gs_ctl** tool
	gs_guc-current.log	Log file that records operations, mainly parameter modification performed by using the **gs_guc** tool

Type	Log File Name	Description
	gaussdbinstall.log	GaussDB installation log file
	gaussdbuninstall.log	GaussDB uninstallation log file
HA script run log	floatip_ha.log	Log file that records the script of floating IP addresses
	gaussDB_ha.log	Log file that records the script of GaussDB resources
	ha_monitor.log	Log file that records the HA process monitoring information
	send_alarm.log	Alarm sending log file
	ha.log	HA run log file
DBService audit log	dbservice_audit.log	Audit log file that records DBService operations, such as backup and restoration operations

Log Level

Table 4-2 describes the log levels supported by DBService. The priorities of log levels are ERROR, WARN, INFO, and DEBUG in descending order. Logs whose levels are higher than or equal to the specified level are printed. The number of printed logs decreases as the specified log level increases.

Table 4-2 Log levels

Level	Description
ERROR	Error information about the current event processing
WARN	Exception information about the current event processing
INFO	Normal running status information about the system and events
DEBUG	System information and system debugging information

Log Format

The following table lists the DBService log formats.

Table 4-3 Log format

Type	Format	Example
Run log	[<*yyyy-MM-dd HH:mm:ss*>] <*Log level*>: [< *Name of the script that generates the log*: *Line number* >]: < *Message in the log*>	[2020-12-19 15:56:42] INFO [postinstall.sh:653] Is cloud flag is false. (main)
Audit log	[<*yyyy-MM-dd HH:mm:ss,SSS*>] UserName:<*Username*> UserIP:<*User IP address*> Operation:<*Operation content*> Result:<*Operation results*> Detail:<*Detailed information*>	[2020-05-26 22:00:23] UserName:omm UserIP:192.168.10.21 Operation:DBService data backup Result: SUCCESS Detail: DBService data backup is successful.

5 Using Doris

5.1 Overview of the Doris Data Model

Basic Concepts

In Doris, data is logically described in the form of tables. A table consists of rows and columns. A row is a row of user data, and a column is used to describe different fields in a row of data. Columns are classified into keys and values. From the service perspective, Key and Value can correspond to dimension columns and metric columns, respectively.

Doris data models are classified into the following types:

- Aggregate
- Unique
- Duplicate

AGGREGATE KEY

The statement for creating an Aggregate model table is as follows:

CREATE TABLE IF NOT EXISTS *example_db.example_tbl*

(

`user_id` LARGEINT NOT NULL COMMENT "User ID",

`date` DATE NOT NULL COMMENT "Data import date and time",

`city` VARCHAR(20) COMMENT "City of the user",

`age` SMALLINT COMMENT "User age",

`gender` TINYINT COMMENT "User Gender",

`last_visit_date` DATETIME REPLACE DEFAULT "1970-01-01 00:00:00" COMMENT " Last access time of the user",

`cost` BIGINT SUM DEFAULT "0" COMMENT "Total consumption",

`max_dwell_time` INT MAX DEFAULT "0" COMMENT "Dwell time",

`min_dwell_time` INT MIN DEFAULT "99999" COMMENT "Minimum dwell time"

)

AGGREGATE KEY(`user_id`, `date`, `city`, `age`, `gender`)

DISTRIBUTED BY HASH(`user_id`) BUCKETS 1

PROPERTIES (

"replication_allocation" = "tag.location.default: 1"

);

NOTE

When data is imported, rows with the same contents in the Key columns will be aggregated into one row, and their values in the Value columns will be aggregated as their **AggregationType** specify. Currently, AggregationType has the following four aggregation modes:

- SUM: Accumulate the values in multiple rows.
- REPLACE: The newly imported value replaces the previous value.
- MAX: Keep the maximum value.
- MIN: Keep the minimum value.

Columns in a table are classified into Key (dimension column) and Value (metric column) based on whether AggregationType is set. For example, if AggregationType is not set, such as user_id, date, and age, is called Key, and if AggregationType is set, is called Value.

UNIQUE KEY model

- Read-on-Read Combination

  These tables do not need to be aggregated. You only need to ensure that the primary keys (user_id and username) are unique. In addition, the read-time combination implementation of the Unique model can be replaced by the REPLACE mode of the Aggregate model. The following is an example:

  CREATE TABLE IF NOT EXISTS *example_db.example_tbl*

  (

  `user_id` LARGEINT NOT NULL COMMENT "User ID",

  `username` VARCHAR(50) NOT NULL COMMENT "Username",

  `city` VARCHAR(20) COMMENT "City of the user",

  `age` SMALLINT COMMENT "User age",

  `gender` TINYINT COMMENT "User Gender",

  `phone` LARGEINT COMMENT "User phone number",

  `address` VARCHAR(500) COMMENT "User address",

  `register_time` DATETIME COMMENT "User registration time"

)

  UNIQUE KEY(`user_id`, `username`)

  DISTRIBUTED BY HASH(`user_id`) BUCKETS 1

  PROPERTIES (

"replication_allocation" = "tag.location.default: 1"

);

- Merge-on-Write

  The Unique model is different from the Aggregate model. The query performance of the Unique model is closer to that of the Duplicate model. Compared with the Aggregate model, the Unique model has great advantages in query performance in scenarios where primary key constraints are required. The Unique model is applicable to aggregation query and query scenarios where a large amount of data needs to be filtered using indexes.

  When creating a table, you can specify the following property to enable the Unique model:

  "enable_unique_key_merge_on_write" = "true"

  For example:

  CREATE TABLE IF NOT EXISTS *example_db.example_tbl*

  (

  `user_id` LARGEINT,

  `username` VARCHAR(50) NOT NULL,

  `city` VARCHAR(20),

  `age` SMALLINT,

  `gender` TINYINT,

  `phone` LARGEINT,

  `address` VARCHAR(500),

  `register_time` DATETIME

)

  UNIQUE KEY(`user_id`, `username`)

  DISTRIBUTED BY HASH(`user_id`) BUCKETS 1

  PROPERTIES (

  "replication_allocation" = "tag.location.default: 1",

  "enable_unique_key_merge_on_write" = "true"

  );

  NOTE

  If the merge-on-write option is enabled for a unique table, the overwritten and updated data is marked and deleted during data import, and new data is written to a new file. During query, all data marked for deletion is filtered at the file level. The read data is the latest data, eliminating the data aggregation process in read-time combination. In addition, multiple predicates can be pushed down, therefore, the performance can be greatly improved in the aggregation query scenario.

Duplicate model

If data does not have a primary key or aggregation requirement, you can use the Duplicate data model to create a table. Duplicate model data is stored based on the data in the imported file and is not aggregated. Even if there are two identical rows of data, they will both be retained. The DUPLICATE KEY specified in the table creation statement is only used to specify that the underlying data is sorted by the specified column.

The statement for creating a duplicate model table is as follows:

CREATE TABLE IF NOT EXISTS *example_db.example_tbl*

(

`timestamp` DATETIME NOT NULL COMMENT " Log time",

`type` INT NOT NULL COMMENT " Log type",

`error_code` INT COMMENT "Error code",

`error_msg` VARCHAR(1024) COMMENT "Error details",

`op_id` BIGINT COMMENT "Owner ID",

`op_time` DATETIME COMMENT "Processing time"

)

DUPLICATE KEY(`timestamp`, `type`, `error_code`)

DISTRIBUTED BY HASH(`type`) BUCKETS 1

PROPERTIES (

"replication_allocation" = "tag.location.default: 1"

);

Key column

For the Duplicate, Aggregate, and Unique models, the Key column is specified during table creation. The differences are as follows:

- Duplicate model: The Key column of a table is only a sequence and does not uniquely identify the table.
- Aggregate and Unique models: For tables of the two aggregation types, the Key column is a real Key column that considers both sorting and unique identifier columns.

Suggestions on Data Model Selection

The data model is determined when the table is created and cannot be modified. Therefore, it is important to select a proper data model.

- The Aggregate model can greatly reduce the amount of data to be scanned and the calculation workload during aggregation query through pre-aggregation. This model is applicable to report query scenarios with fixed modes. However, this model is not applicable to count(*) query. In addition, because the aggregation function in Value columns is fixed, semantic correctness needs to be considered when aggregation queries using other functions are performed.
- The Unique model ensures that the primary key is unique in scenarios where a unique primary key constraint is required. However, the query advantages brought by pre-aggregation such as ROLLUP cannot be used.
  - For users who have high performance requirements for aggregate query, merge-on-write is recommended.

  - The Unique model supports only the update of an entire row. If you need to update both the unique primary key constraint and some columns (for example, in the scenario where multiple source tables are imported to one Doris table), you can use the Aggregate model and set the aggregation type of non-primary key columns to REPLACE_IF_NOT_NULL.
- Duplicate is suitable for ad-hoc query in any dimension. Although the pre-aggregation feature cannot be used, it is not restricted by the aggregation model. The advantages of the column-store model can be brought into full play (only related columns are read, and all key columns do not need to be read).

5.2 Managing Doris User Permissions

5.2.1 About Doris User Permissions

Table 5-1 lists the permissions supported by the Doris.

Table 5-1 Doris Permission List

Permission	Permission Introduction
Node_priv	Node change permission. Add, delete, and bring offline FE, BE, and DBroker nodes. This permission can be granted only to the Global level.
Admin_priv	All permissions except NODE_PRIV.
Grant_priv	Permission change permission, including granting permissions, revoking permissions, and adding, deleting, and changing users and roles. Users with this permission cannot grant the node_priv permission to other users unless they already have the node_priv permission.
Select_priv	Read-only permission on databases and tables.
Load_priv	Write permission on databases and tables, including Load, Insert, and Delete.
Alter_priv	Permission to modify databases and tables. including renaming databases or tables, adding, deleting, or changing columns, and adding or deleting partitions.
Create_priv	Permission to create databases, tables, and views.
Drop_priv	Delete permissions on databases, tables, and views.
Usage_priv	Permissions to use resources and workload groups.

Database table permissions are classified into the following four levels based on the permission application scope:

- CATALOG LEVEL: data directory-level permission. The granted permission applies to any database table in the specified catalog.
- DATABASE LEVEL: database-level permission. The granted permissions apply to any table in the specified database.
- TABLE LEVEL: table-level permission. The granted permissions apply to the specified table in the specified database.
- RESOURCE LEVEL: resource-level permission. The granted permission applies to the specified resource.

5.2.2 Creating a Doris Permission Role

The Doris permission management system implements row-level fine-grained permission control and role-based permission access control.

NOTE

- Only clusters of MRS 3.3.0 and later versions support role assignment on FusionInsight Manager. If the cluster is of MRS 3.3.0 or earlier, you need to connect to the database as user **root** (the default password is empty) regardless of whether Kerberos authentication is enabled.
- In MRS 3.5.0 and later versions, if Kerberos authentication is disabled for the cluster (the cluster is in normal mode), you can also create a user on FusionInsight Manager and bind the user to a role to obtain the required permission.

Prerequisite

- The Doris service is running properly.
- The role name cannot be operator or admin.
- When Kerberos authentication is enabled for the cluster (the cluster is in security mode), it takes about 2 minutes for the permission to take effect after the Doris permission is successfully assigned.

Adding the Doris Role

Step 1 Log in to FusionInsight Manager and choose **System** > **Permission** > **Role**. On the displayed page, click **Create Role**.

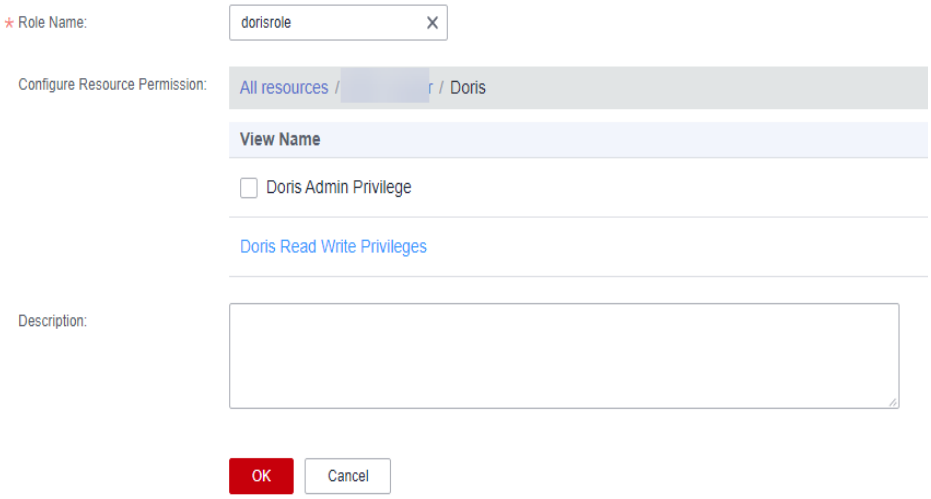
Step 2 Specify **Role Name**. In the **Configure Resource Permission** area, click the cluster name. On the displayed service list page, click the **Doris** service.

Determine whether to create a role with the Doris administrator rights based on service requirements.

NOTE

- The Doris administrator has all the rights except the node operation rights.
- Role Name: The name of the role to be added cannot contain hyphens (-) and cannot start with a digit.

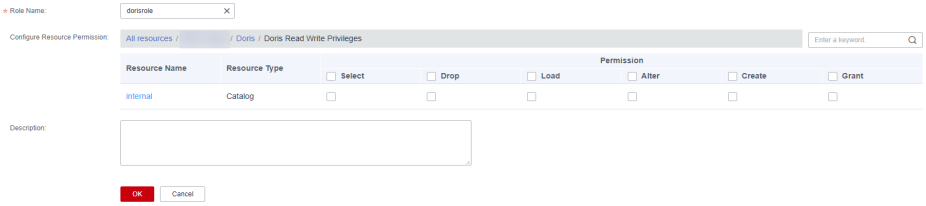
- If yes, go to **Step 3**.
- If no, go to **Step 4**.



Step 3 Select **Doris Admin Privilege** and click **OK**.

Step 4 Click **Doris Read Write Privileges** and select **Select**, **Drop**, **Load**, **Alter**, **Create**, or **Grant** for the corresponding resource.

Determine whether to grant the permission based on the service requirements.



Step 5 After the authorization is complete, click **OK**.

----End

Adding a User and Binding the User to the Doris Role

Step 1 Log in to FusionInsight Managerr and choose **System** > **Permission** > **User** and click **Create**.

- **Username**: Enter the name of the user you want to create.
- **User Type**:
  - If Kerberos authentication is enabled for the cluster (the cluster is in security mode), select **Human-Machine**.
  - If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), select **Machine-Machine**.
- **Password** and **Confirm New Password**: Enter a password for the human-machine user and enter it again.
- **Role**: Click **Add**. In the displayed dialog box, select a role with the Doris permission and click **OK**.

NOTE

- Username: The username to be added cannot contain hyphens (-). Otherwise, the authentication fails.
- Password: The password cannot contain special characters $, ., and #. Otherwise, the authentication will fail.

Step 2 Click **OK**.

Step 3 Use the newly created user to connect to the Doris service.

- Kerberos authentication is enabled for the cluster (the cluster is in security mode)
  a. Log in to FusionInsight Manager as the new human-machine user and change the initial password.
  b. Log in to the node where the MySQL client is installed and use the new username and new password to connect to the Doris service.

     export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

     mysql -u*dorisuser* **-P -P***Connection port for FE queries* **-h***IP address of the Doris FE instance*

     Enter the password for logging in to the database.

     NOTE

     - To obtain the query connection port of the Doris FE instance, you can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
     - To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the service IP address of any FE instance.
     - You can also use the MySQL connection software or Doris web UI to connect to the database.
- Kerberos authentication is disabled for the cluster (the cluster is in normal mode)
  a. Log in to the node where the MySQL client is installed and connect to the Doris service as user **admin**.

     mysql -uadmin-P*Connection port for FE queries* **-h***IP address of the Doris FE instance*

     NOTE

     The default password of user **admin** is empty.
  b. Set a password for the machine-machine user.

     SET PASSWORD FOR '*Machine-Machine Username***' = PASSWORD('***xxx***')**
  c. Connect to Doris on the MySQL client using the machine-machine user.

     mysql -u*Machine-machine username* **-p-P***Connection port for FE queries* **-h***IP address of the Doris FE instance*

     Enter the password of the user.

----End

Adding a Role and Binding It to a User (Kerberos authentication is disabled for the cluster (the cluster is in normal mode))

This operation applies only to versions earlier than MRS 3.5.0.

Step 1 Log in to the node where the MySQL client is installed and connect to the Doris service as user **admin**.

mysql -uadmin -P*Database connection port* **-h***IP address of Doris FE instance*

NOTE

- The default password of user **admin** is empty.
- The database connection port is the query connection port of the Doris FE. You can also log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris WebUI to connect to the database.

Step 2 Run the following command to create a role:

CREATE ROLE *dorisrole***;**

Step 3 Run the following command to grant permissions to the role. For details about the permissions, see **About Doris User Permissions**. For example, to grant the ADMIN_PRIV permission to the role, run the following command:

GRANT *ADMIN_PRIV* **ON *.*.* TO ROLE '***dorisrole***';**

Step 4 Create a user.

- Create a user and grant the user a role.

  CREATE USER '*dorisuser***'@'%' IDENTIFIED BY '***password***' DEFAULT ROLE '***dorisrole***';**

- Create a user, add a password expiration policy, and grant the user a role.

  For example, run the following commands to create the **dorisuser** user, set the user password to expire in 180 days, and grant the user the **dorisrole**:

  CREATE USER '*dorisuser***'@'%' IDENTIFIED BY '***password***' PASSWORD_EXPIRE INTERVAL** *180 DAY***;**

  GRANT '*dorisrole***' TO '***dorisuser***'@'%';**

  NOTE

  - If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), the user password never expires by default, which poses security risks. You are advised to set a password expiration policy to request the user to change the password periodically.
  - After the password expires, the user is unavailable. You must change the password as a user with the same level of rights as the Doris user.

----End

5.2.3 Column Permission Management

Scenarios

You can manage column-level permission in Doris by adding the **enable_col_auth** parameter to the custom configuration of the FE service.

NOTE

- This feature is available for MRS 3.3.1 or later only.
- Only the **Select_priv** permission is supported by this function.
- You must use a user with the **Grant_priv** permission to manage column permission.
- If a user with the column-level **Select_priv** permission runs **select *** to query table data, it can access only the columns allowed by the permission.
- If a user with the column-level **Select_priv** permission runs **desc** *tbl* to query table details, it can access only the information of columns allowed by the permission.
- Column-level permission is also available for views and materialized views.

Prerequisite

- A cluster containing the Doris service has been created, and all services in the cluster are running properly.
- The nodes to be connected to the Doris database can communicate with the MRS cluster.
- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.

Procedure

Step 1 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris**, and click **Configurations** and then **All Configurations**.

Step 2 Choose **FE(Role)** > **Customization**. Enter the custom parameter **enable_col_auth**, set its value to **true**, and add it to the **fe.conf** file.

Step 3 Click **Save** and then **OK**.

Step 4 Click **Instances**, select all FE instances, and choose **More** > **Restart Instance**.

Step 5 Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login user* **-p -P***Connection port for FE queries* **-h***IP address of the Doris FE instance*

Enter the password for logging in to the database.

NOTE

- To obtain the query connection port of the Doris FE instance, you can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the service IP address of any FE instance.
- You can also use the MySQL connection software or Doris web UI to connect to the database.

Step 6 Run the following commands to grant the **Select_priv** permission:

- Grant permission to a user.
  GRANT select_priv [(col1, col2...)] ON *ctl.db.tbl* **TO '***user***';**
- Grant permission to a role.
  GRANT select_priv [(col1, col2...)] ON *ctl.db.tbl* **TO ROLE '***role***';**

Step 7 Check the user permission.

show grants for *user***;**

Step 8 Revoke the **Select_priv** permission.

- Revoke permission from a user.
  revoke select_priv [(col1, col2...)] ON *ctl.db.tbl* **from '***user***';**
- Revoke permission from a role.
  revoke select_priv [(col1, col2...)] ON *ctl.db.tbl* **from ROLE '***role***';**

Step 9 Check user permission.

show grants for *user***;**

----End

5.3 Using the MySQL Client to Connect to Doris

Doris supports the MySQL protocol. Therefore, most clients that support the MySQL protocol can access Doris, including the CLI or IDE, such as MariaDB, DBeaver, and Navicat for MySQL.

This section uses the MySQL 8.0.22 client to connect to Doris as an example.

Prerequisite

The node where the MySQL client is to be installed can communicate with the MRS cluster.

Procedure

Step 1 Log in to the node where the MySQL client is to be installed as the root user.

Step 2 Run the following command to check the version of the ncurses-libs library on which the MySQL client depends:

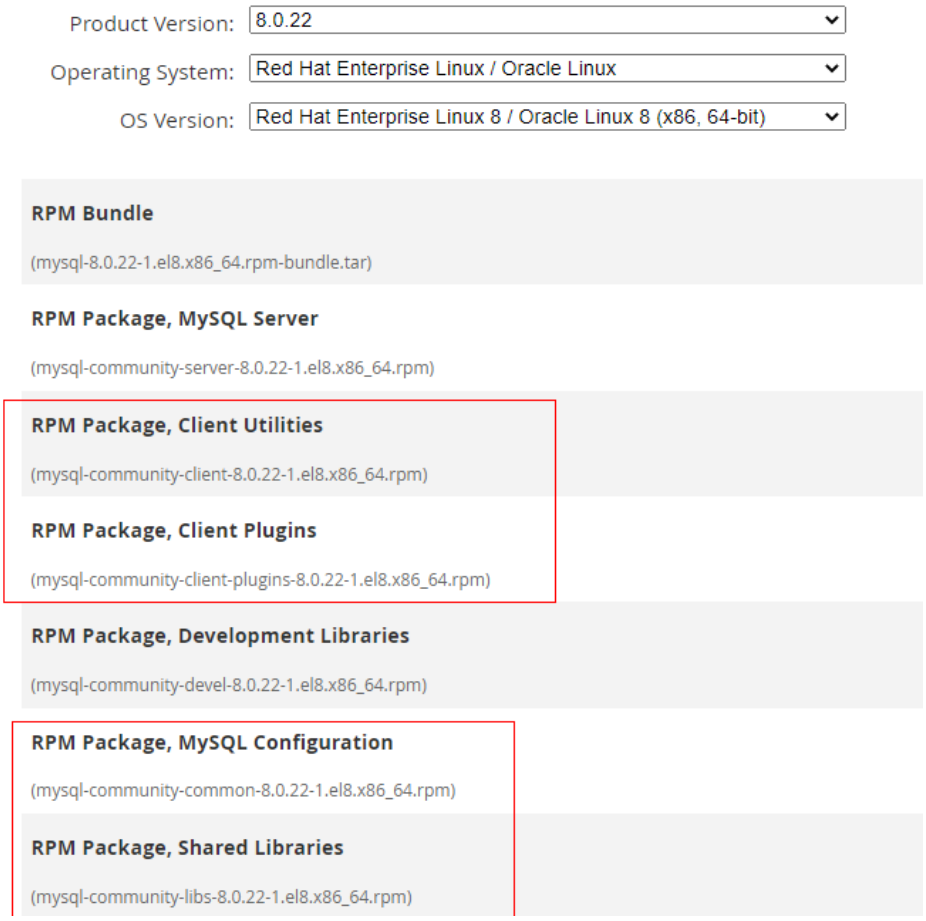
rpm -qa | grep ncurses


```
[root@node-master1hlrt ~]# rpm -qa | grep ncurses
ncurses-base-6.3-2.r2.hce2.noarch
ncurses-libs-6.3-2.r2.hce2.x86_64
ncurses-6.3-2.r2.hce2.x86_64
```

Step 3 Download the software package corresponding to the MySQL client from **https://downloads.mysql.com/archives/community/**. You are advised to install MySQL 8.x. The following uses Red Hat as an example:

- If the dependent library in **Step 2** is 6.x, you are advised to download the MySQL software package whose OS version is Red Hat 8.
- If the dependent library in **Step 2** is 5.x, you are advised to download the MySQL software package whose OS version is Red Hat 7.

For example, to install the MySQL 8.0.22 client of Red Hat, download the following software packages:



Step 4 Upload the downloaded software package to the node where the MySQL client is to be installed.

Step 5 Run the following command in the directory where the uploaded file is stored to install the MySQL client and the corresponding dependency package:

rpm -ivh mysql-community-client-8.0.22-1.el8.x86_64.rpm --nodeps --force

rpm -ivh mysql-community-client-plugins-8.0.22-1.el8.x86_64.rpm --nodeps --force

rpm -ivh mysql-community-common-8.0.22-1.el8.x86_64.rpm --nodeps --force

rpm -ivh mysql-community-libs-8.0.22-1.el8.x86_64.rpm --nodeps --force

Step 6 Run the following command to check the MySQL client version:

mysql --version

```
[root@node-master1hlrt ~]# mysql --version
mysql  Ver 8.0.22 for Linux on x86_64 (MySQL Community Server - GPL)
```

Step 7 After the MySQL client is successfully installed, you can access Doris. For details, see **Getting Started with Doris**.

----End

5.4 Getting Started with Doris

Doris is a high-performance and real-time analytical database based on the MPP architecture. It supports not only high-concurrency point query scenarios, but also high-throughput complex analysis scenarios.

This document uses examples to describe how to use an MRS Doris cluster to perform basic table creation and query operations.

NOTE

Doris database names and table names are case sensitive.

Prerequisite

- A cluster containing the Doris service has been created, and all services in the cluster are running properly.
- The node to be connected to the Doris database can communicate with the MRS cluster.
- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.

Procedure

Step 1 Create a user with the Doris management permission.

- Kerberos authentication is enabled for the cluster (the cluster is in security mode)
  a. Log in to FusionInsight Manager and choose **System**. In the navigation pane on the left, click **Permission** > **Role**, and click **Create Role**. On the displayed page, enter the role name, for example, **dorisrole**. In the **Configure Resource Permission** area, select *target cluster* > **Doris**, select **Doris Admin Privilege**, and click **OK**.
  b. Choose **User** > **Create**, enter a username, for example, **dorisuser**, set **User Type** to **Human-Machine**, retain the **default** value for **Password Policy**, enter the user password, confirm the password, associate the user with the **dorisrole** role, and click **OK**.

c. Log in to FusionInsight Manager as the new dorisuser user and change the initial password of the user.

- Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

  a. Log in to the node where the MySQL client is installed and connect to the Doris service as user **admin**.

     mysql -uadmin -P*Database connection port* **-h***IP address of Doris FE instance*

     NOTE

     - The default password of user **admin** is empty.
     - The database connection port is the query connection port of the Doris FE. You can also log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
     - To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
     - You can also use the MySQL connection software or Doris WebUI to connect to the database.
     - Only clusters of MRS 3.3.0 and later versions with Kerberos authentication enabled support role assignment on FusionInsight Manager. If the cluster is of MRS 3.3.0 or earlier, you need to connect to the database as user **root** (the default password is empty) regardless of whether Kerberos authentication is enabled.

  b. Run the following command to create a role:

     CREATE ROLE *dorisrole***;**

  c. Run the following command to grant permissions to the role. For details about the permissions, see **About Doris User Permissions**. For example, to grant the ADMIN_PRIV permission to the role, run the following command:

     GRANT *ADMIN_PRIV* **ON *.*.* TO ROLE '***dorisrole***';**

  d. Run the following commands to create a user and bind the user to a role:

     CREATE USER '*dorisuser***'@'%' IDENTIFIED BY '***password***' DEFAULT ROLE '***dorisrole***';**

     There can be security risks if a command contains the authentication password. You are advised to disable the command recording function (history) before running the command.

Step 2 Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login user* **-p***Database login user password* **-P***Database connection port* **-h***Doris FE instance IP address*

NOTE

- The database connection port is the query connection port of the Doris FE. You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris WebUI to connect to the database.

Step 3 Run the following command to check the running status of the FE:

SHOW FRONTENDS\G;

SHOW BACKENDS\G;

Step 4 Create a database.

create database if not exists *mrs_demo*;

use *mrs_demo*;

NOTE

For more information about Doris SQL commands and syntax, see the **Doris SQL Manual**.

Step 5 Run the following command to create a data table in the database:

CREATE TABLE IF NOT EXISTS *mrs_table*

(

`user_id` LARGEINT NOT NULL COMMENT "User ID",

`date` DATE NOT NULL COMMENT " Data Import Date",

`city` VARCHAR(20) COMMENT "city",

`age` SMALLINT COMMENT "age",

`gender` TINYINT COMMENT "Gender",

`last_visit_date` DATETIME REPLACE DEFAULT "1970-01-01 00:00:00" COMMENT " Last access time of the user",

`cost` BIGINT SUM DEFAULT "0" COMMENT "Total consumption",

`max_dwell_time` INT MAX DEFAULT "0" COMMENT "Dwell time",

`min_dwell_time` INT MIN DEFAULT "99999" COMMENT "Minimum dwell time"

)

AGGREGATE KEY(`user_id`, `date`, `city`, `age`, `gender`)

DISTRIBUTED BY HASH(`user_id`) BUCKETS 1

PROPERTIES (

"replication_allocation" = "tag.location.default: 1"

);

Step 6 Create the test.csv file in any directory on the current node. The file content is as follows:

```
10000,2017-10-01,city1,20,0,2017-10-01 06:00:00,20,10,10
10000,2017-10-01,city2,20,0,2017-10-01 07:00:00,15,2,2
10001,2017-10-01,city3,30,1,2017-10-01 17:05:45,2,22,22
10002,2017-10-02,city4,20,1,2017-10-02 12:59:12,200,5,5
10003,2017-10-02,city5,32,0,2017-10-02 11:20:00,30,11,11
10004,2017-10-01,city6,35,0,2017-10-01 10:00:15,100,3,3
10004,2017-10-03,city7,35,0,2017-10-03 10:20:22,11,6,6
```

Step 7 Import data in the test.csv file to the table created in **Step 5** using Stream Load.

cd Directory where test.csv is stored

curl -k --location-trusted -u *doris User name*:*User password* **-H "***label:table1_20230217***" -H "***column_separator:,***" -T test.csv http://***Doris FE Instance IP address*:*HTTP port*/api/mrs_demo/mrs_table/_stream_load**

NOTE

- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- To view the HTTP port number, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and search for **http_port**.

Step 8 Run the following command to query the data:

select * from *mrs_table* **where city='city1';**

----End

5.5 Importing Doris Data

5.5.1 Importing Data to Doris with Broker Load

Broker load is an asynchronous import method, and the supported data sources depend on the data sources supported by the Broker process.

Data in the Doris table is ordered. When importing data, Broker Load uses Doris cluster resources to sort data. Compared with Spark Load which is used to migrate massive historical data, Broker Load occupies a large number of Doris cluster resources. The Broker Load mode is used when users do not have Spark compute resources. If Spark compute resources are available, Spark Load is recommended.

You need to import data with Broker Load using MySQL protocol and check the import result by viewing the import command. Broker Load is used for the following scenarios:

- The source data is in a storage system that the broker can access, such as HDFS.
- The data volume ranges from tens to hundreds of GB.
- Data in CSV, Parquet, and ORC formats can be imported. Only data in CSV format is supported by default.

Prerequisites

- A cluster containing the Doris service has been created, and all services in the cluster are running properly.
- The node to be connected to the Doris database can communicate with the MRS cluster.
- A user with Doris management permission has been created.
  - Kerberos authentication is enabled for the cluster (the cluster is in security mode)

    Log in to FusionInsight Manager, create a human-machine user, for example, **dorisuser**, create a role with Doris administrator permissions, and bind the role to the user.

    Log in to FusionInsight Manager as the created **dorisuser** user, and change the initial password.
  - Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

    After connecting to Doris as user **admin**, create a role with administrator permissions, and bind the role to the user.
- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.
- The DBroker instance has been installed and started in Doris.
- The Hive client has been installed.
- If Doris imports data across clusters through Broker Load, you need to configure cross-cluster mutual trust. For details, see **Configuring Cross-Manager Mutual Trust Between Clusters**.

Importing Hive Table Data to Doris

- Import Hive table data in Text format to Doris.
  a. Log in to the node where the Hive client is installed as the client installation user and run the following commands to log in to the Hive beeline CLI:

     cd *Client installation directory*

     source bigdata_env

     kinit *Component service user* (If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), skip this step.)

  b. Run the following statements to create a Hive table in the **default** database (the partition field is **c4**):

     CREATE TABLE *test_table***(**

     `c1` int,

     `c2` int,

     `c3` string)

     PARTITIONED BY (c4 string)

     row format delimited fields terminated by ','lines terminated by '\n' stored as textfile ;

  c. Run the following command to insert data to the Hive table:

insert into table *test_table* **values(1,1,'1','2022-04-10'), (2,2,'2','2022-04-22');**

d. Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login user* **-p***Database login user password* **-P***Database connection port* **-h***IP address of Doris FE instance*

NOTE

- The database connection port is the query connection port of the Doris FE. You can also log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris WebUI to connect to the database.
- If the Hive and Doris components are deployed across clusters, modify the following configurations:
  - The value of **hadoop.rpc.protection** of Doris in the cluster where Doris is located must be the same as that of HDFS in the cluster where Hive is located.
  - Change the value of **-Djava.security.krb5.conf** in configuration item **BROKER_GC_OPTS** of DBRoker in the cluster where Doris is located. Copy the **$BIGDATA_HOME/FusionInsight_HD_*/*_HiveServer/etc/kdc.conf** file of any HiveServer node in the Hive cluster to any directory of any DBRoker node in the Doris cluster.

e. Run the following command to create a Doris table:

```
CREATE TABLE example_db.test_t1 (
`c1` int NOT NULL,
`c4` date NULL,
`c2` int NOT NULL,
`c3` String NOT NULL
) ENGINE=OLAP
UNIQUE KEY(`c1`, `c4`)
PARTITION BY RANGE(`c4`)
(
PARTITION P_202204 VALUES [('2022-04-01'), ('2022-05-01')))
DISTRIBUTED BY HASH(`c1`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 3",
"dynamic_partition.enable" = "true",
"dynamic_partition.time_unit" = "MONTH",
"dynamic_partition.start" = "-2147483648",
```


"dynamic_partition.end" = "2",

"dynamic_partition.prefix" = "P_",

"dynamic_partition.buckets" = "1",

"in_memory" = "false",

"storage_format" = "V2"

);

f. Run the following command to import data:

- Kerberos authentication is enabled for the cluster (the cluster is in security mode)

  LOAD LABEL *broker_load_2022_03_23*

  (

  DATA INFILE("hdfs://IP address of the active NameNode instance:RPC port number/user/hive/warehouse/test_table/*/*")

  INTO TABLE *test_t1*

  COLUMNS TERMINATED BY ","

  (c1,c2,c3)

  COLUMNS FROM PATH AS (`c4`)

  SET

  (

  c4 = str_to_date(`c4`,'%Y-%m-%d'),c1=c1,c2=c2,c3=c3

)

)

  WITH BROKER "*broker_192_168_67_78***"**

  (

  "hadoop.security.authentication"="kerberos",

  "kerberos_principal"="doris/hadoop.hadoop.com@HADOOP.COM",

  "kerberos_keytab"="${BIGDATA_HOME}/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-fe/bin/doris.keytab"

)

  PROPERTIES

  (

  "timeout"="1200",

  "max_filter_ratio"="0.1"

  );

- Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

  LOAD LABEL *broker_load_2022_03_23*

  (

  DATA INFILE("hdfs://*IP address of the active NameNode instance***:***RPC port number***/user/hive/warehouse/***test_table***/*/*")**

```
INTO TABLE test_t1
COLUMNS TERMINATED BY ","
(c1,c2,c3)
COLUMNS FROM PATH AS (`c4`)
SET
(
c4 = str_to_date(`c4`,'%Y-%m-%d'),c1=c1,c2=c2,c3=c3
)
)
WITH BROKER "broker_192_168_67_78"
(
"username"="hdfs",
"password"=""
)
PROPERTIES
(
"timeout"="1200",
"max_filter_ratio"="0.1"
);
```

NOTE

- To view the IP address of the active NameNode instance, log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS** > **Instances**.
- You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** > **Configurations**, and search for **dfs.namenode.rpc.port** to view the RPC port number.
- *broker_192_168_67_78* indicates the broker name. You can run the **show broker;** command on the MySQL client to view the broker name.

g. Run the following command to check the status of the import task:

show load order by createtime desc limit 1\G;

```
JobId: 41326624
Label: broker_load_2022_03_23
State: FINISHED
Progress: ETL:100%; LOAD:100%
Type: BROKER
EtlInfo: unselected.rows=0; dpp.abnorm.ALL=0; dpp.norm.ALL=27
TaskInfo: cluster:N/A; timeout(s):1200; max_filter_ratio:0.1
ErrorMsg: NULL
CreateTime: 2022-04-01 18:59:06
EtlStartTime: 2022-04-01 18:59:11
EtlFinishTime: 2022-04-01 18:59:11
LoadStartTime: 2022-04-01 18:59:11
LoadFinishTime: 2022-04-01 18:59:11
URL: NULL
JobDetails: {"Unfinished backends":{"5072bde59b74b65-8d2c0ee5b029adc0":
[]},"ScannedRows":27,"TaskNumber":1,"All backends":{"5072bde59b74b65-8d2c0ee5b029adc0":
[36728051]},"FileNumber":1,"FileSize":5540}
1 row in set (0.01 sec)
```

h. You can manually cancel an import task whose Broker Load job status is not **CANCELLED** or **FINISHED**. To cancel an import task, you need to specify the label of the import task. The statement is as follows:

CANCEL LOAD FROM *Database name* WHERE LABEL = "*Label name*";

For example, to cancel the import job whose label is **broker_load_2022_03_23** in database **demo**, run the following command:

CANCEL LOAD FROM demo WHERE LABEL = "broker_load_2022_03_23";

- Importing Hive Table Data in ORC Format to Doris

  a. Log in to the node where the Hive client is installed as the client installation user and run the following commands to log in to the Hive beeline CLI:

  cd *Client installation directory*

  source bigdata_env

  kinit *Component service user* (If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), skip this step.)

  b. Run the following command to create a Hive table in ORC format in the **default** database:

  CREATE TABLE *test_orc_tbl*(

  `c1` int,

  `c2` int,

  `c3` string)

  PARTITIONED BY (c4 string)

  row format delimited fields terminated by ','lines terminated by '\n' stored as orc;

  c. Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

  If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

  export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

  mysql -u*Database login user* **-p***Database login user password* **-P***Database connection port* **-h***IP address of Doris FE instance*

NOTE

- The database connection port is the query connection port of the Doris FE. You can also log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris WebUI to connect to the database.
- If the Hive and Doris components are deployed across clusters, modify the following configurations:
  - The value of **hadoop.rpc.protection** of Doris in the cluster where Doris is located must be the same as that of HDFS in the cluster where Hive is located.
  - Change the value of **-Djava.security.krb5.conf** in configuration item **BROKER_GC_OPTS** of DBRoker in the cluster where Doris is located. Copy the **$BIGDATA_HOME/FusionInsight_HD_*/*_HiveServer/etc/kdc.conf** file of any HiveServer node in the Hive cluster to any directory of any DBRoker node in the Doris cluster.

d. Run the following command to create a Doris table:

CREATE TABLE *example_db.test_orc_t1* **(**
`c1` int NOT NULL,
`c4` date NULL,
`c2` int NOT NULL,
`c3` String NOT NULL
) ENGINE=OLAP
UNIQUE KEY(`c1`, `c4`)
PARTITION BY RANGE(`c4`)
(
PARTITION P_202204 VALUES [('2022-04-01'), ('2022-05-01')))
DISTRIBUTED BY HASH(`c1`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 3",
"dynamic_partition.enable" = "true",
"dynamic_partition.time_unit" = "MONTH",
"dynamic_partition.start" = "-2147483648",
"dynamic_partition.end" = "2",
"dynamic_partition.prefix" = "P_",
"dynamic_partition.buckets" = "1",
"in_memory" = "false",
"storage_format" = "V2"
);

e. Run the following command to import data using Broker Load:

- Kerberos authentication is enabled for the cluster (the cluster is in security mode):

  LOAD LABEL *broker_load_2022_03_24*

  (

  DATA INFILE("hdfs://*IP address of the active NameNode instance*:*RPC port number*/**user/hive/warehouse/***test_orc_tbl*/***/*")**

  INTO TABLE *test_orc_t1*

  FORMAT AS "orc"

  (c1,c2,c3)

  COLUMNS FROM PATH AS (`c4`)

  SET

  (

  c4 = str_to_date(`c4`,'%Y-%m-%d'),c1=c1,c2=c2,c3=c3

)

)

  WITH BROKER "broker_192_168_67_78"

  (

  "hadoop.security.authentication"="kerberos",

  "kerberos_principal"="doris/hadoop.hadoop.com@HADOOP.COM",

  "kerberos_keytab"="${BIGDATA_HOME}/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-fe/bin/doris.keytab"

)

  PROPERTIES

  (

  "timeout"="1200",

  "max_filter_ratio"="0.1"

  );

- Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

  LOAD LABEL *broker_load_2022_03_24*

  (

  DATA INFILE("hdfs://*IP address of the active NameNode instance*:*RPC port number*/**user/hive/warehouse/***test_orc_tbl*/***/*")**

  INTO TABLE *test_orc_t1*

  FORMAT AS "orc"

  (c1,c2,c3)

  COLUMNS FROM PATH AS (`c4`)

  SET

  (

  c4 = str_to_date(`c4`,'%Y-%m-%d'),c1=c1,c2=c2,c3=c3


```
)
)
WITH BROKER "broker_192_168_67_78"
(
'username"="hdfs",
'password"=""
)
PROPERTIES
(
"timeout"="1200",
"max_filter_ratio"="0.1"
);
```

NOTE

- **FORMAT AS "orc"** : The format of the data to be imported is ORC.
- **SET**: The field mapping between Hive tables and Doris tables and field conversion rules.
- To view the IP address of the active NameNode instance, log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS** > **Instances**.
- You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** > **Configurations**, and search for **dfs.namenode.rpc.port** to view the RPC port number.
- *broker_192_168_67_78* indicates the broker name. You can run the **show broker;** command on the MySQL client to view the broker name.

f. Run the following statement to check the status of the imported task:

show load order by createtime desc limit 1\G;

g. You can manually cancel an import task whose Broker Load job status is not **CANCELLED** or **FINISHED**. To cancel an import task, you need to specify the label of the import task. The statement is as follows:

CANCEL LOAD FROM *Database name* WHERE LABEL = "*Label name*";

For example, to cancel the import job whose label is **broker_load_2022_03_23** in database **demo**, run the following command:

CANCEL LOAD FROM demo WHERE LABEL = "broker_load_2022_03_23";

Related Parameter Configurations

The following configurations are system-level configurations of Broker Load and apply to all Broker Load import tasks.

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations** > **FE (Role)** > **Customization**, and add the following parameters to the customized parameter fe.conf.customized.configs:

- **min_bytes_per_broker_scanner**: specifies the minimum amount of data that can be processed by a single BE. The default value is **64 MB**.
- **max_bytes_per_broker_scanner**: specifies the maximum amount of data that can be processed by a single BE. The default value is **3 GB**.

- **max_broker_concurrency**: specifies the maximum number of concurrent import tasks in a job. The default value is **10**.

The minimum amount of data to be processed, maximum number of concurrent tasks, source file size, and number of BE nodes in the current cluster determine the number of concurrent tasks to be imported.

- Number of concurrent import tasks = Math.min (Source file size/Minimum processing volume, Maximum number of concurrent import tasks, Number of current BE nodes)
- Processing volume of a single BE = Size of the source file/Number of concurrent import tasks

Usually the maximum amount of data supported by an import job is the product of the value of **max_bytes_per_broker_scanner** and the number of BE nodes. To import a larger amount of data, you need to adjust the value of **max_bytes_per_broker_scanner**.

5.5.2 Importing OBS Data to Doris with Broker Load

To import data to Doris with Stream Load, you need to read the data first by the client. Broker Load simplifies this process by sending import requests to Doris, and Doris pulls data. If the data you want to import is stored in object storage, Broker Load is the most convenient tool. With Broker Load, data is directly read and imported by Doris without passing through the client.

You need to import data with Broker Load through MySQL protocol and check the import result with the import command. Broker Load is suitable for the following scenes:

- The source data is in a storage system that the broker can access, such as OBS.
- The amount of data is at the level of tens to hundreds of GB.
- Data in CSV, Parquet, ORC, and JSON formats can be imported. Only data in CSV format is supported by default.

The operations in this section are applicable to MRS 3.5.0 and later versions.

Prerequisites

- A cluster containing the Doris service has been created, and all services in the cluster are running properly.
- The nodes to be connected to the Doris database can communicate with the MRS cluster.
- A user with Doris management permission has been created.
  - Kerberos authentication is enabled for the cluster (the cluster is in security mode)

    Log in to FusionInsight Manager, create a human-machine user, for example, **dorisuser**, create a role with Doris administrator permissions, and bind the role to the user.

    Log in to FusionInsight Manager as the new user **dorisuser** and change the initial password.

  - Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

    After connecting to Doris as user **admin**, create a role with administrator permissions, and bind the role to the user.

- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.
- You have prepared the data files to be imported to Doris.

Creating an OBS Parallel File System and Obtaining the AK/SK

Create an OBS parallel file system.

Step 1 Log in to the OBS console.

Step 2 Choose **Parallel File Systems** > **Create Parallel File System**.

Step 3 Enter a file system name, for example, **doris-obs**.

The name of an enterprise project must be the same as that of the MRS cluster. Set other parameters.

Step 4 Click **Create Now**.

Step 5 In the parallel file system list, click the name of the one you just created and click **Overview** to obtain the endpoint information.

NOTE

After a service is deleted or a cluster is uninstalled, dirty data may remain in the parallel file system created in **Step 2** to **Step 4**. You need to delete the dirty data.

Step 6 Click **Files** in the navigation pane, click **Create Folder**, enter a folder name, for example, **test**, and click **OK**.

Step 7 Click the name of the new folder and click **Upload File** to upload the data, for example, to **test_data.csv**.

Obtain AK/SK information.

Step 8 Move the cursor on the username in the upper right corner, and select **My Credentials** from the drop-down list.

Step 9 Click **Access Keys**, click **Create Access Key**, and enter the verification code or password. Click **OK** and download the access keys. Obtain the AK/SK information from the **.csv** file.

----End

Importing OBS Data to a Doris Table

Step 1 Log in to the node where MySQL is installed and connect the Doris database.

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login user* **-p -P***Connection port for FE queries* **-h***IP address of the Doris FE instance*

Enter the password for logging in to the database.

NOTE

- To obtain the query connection port of the Doris FE instance, you can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris web UI to connect the database.

Step 2 Create a database.

create database *test_broker_load*;

use *test_broker_load*;

Step 3 Create a table and import OBS data to the table.

CREATE TABLE IF NOT EXISTS *test*

(

`user_id` LARGEINT NOT NULL COMMENT "User ID",

`city` VARCHAR(20) COMMENT "City",

`age` SMALLINT COMMENT "Age",

`gender` TINYINT COMMENT "Gender",

`cost` BIGINT SUM DEFAULT "0" COMMENT "Total consumption",

`max_dwell_time` INT MAX DEFAULT "0" COMMENT "Maximum dwell time",

`min_dwell_time` INT MIN DEFAULT "99999" COMMENT "Minimum dwell time"

)

AGGREGATE KEY(`user_id`, `city`, `age`, `gender`)

DISTRIBUTED BY HASH(`user_id`) BUCKETS 100

PROPERTIES (

"replication_allocation" = "tag.location.default: 3"

);

LOAD LABEL *brokerload_test_label001*

(

DATA INFILE("obs://*Parallel file system name***/test/***test_data.csv***")**

INTO TABLE `***test***`

COLUMNS TERMINATED BY ','

FORMAT AS "*csv***"**

)

WITH BROKER "broker1"

(

"fs.obs.access.key" = "*xxx*",

"fs.obs.secret.key" = "*xxx*",

"fs.obs.endpoint" = "*xxx*"

);

NOTE

- **LOAD LABEL**: unique label that must be specified for each import task. You can use this label to view the job progress.
- **DATA INFILE**: OBS path of the data files (uploaded in **Step 7**) you want to import to Doris.
- **COLUMNS TERMINATED BY**: column separator. This parameter is required only when the CSV format is used. Only a single-byte separator can be specified.
- **FORMAT AS**: file type. The value can be **CSV**, **JSON**, **PARQUET**, or **ORC**. The default value is CSV.
- **WITH BROKER**: name of the broker service to be used. You can run the following command to view the broker information of the cluster:

  show broker;

  ```
  mysql> show broker;
  +----------------+----------+-------+-------+---------------------+---------------------+--------+
  | Name           | Host     | Port  | Alive | LastStartTime       | LastUpdateTime      | ErrMsg |
  +----------------+----------+-------+-------+---------------------+---------------------+--------+
  | broker_[redacted] | [redacted] | 29990 | true  | 2024-07-26 17:19:21 | 2024-07-29 17:44:05 |        |
  | broker_[redacted] | [redacted] | 29990 | true  | 2024-07-26 17:19:21 | 2024-07-29 17:44:05 |        |
  +----------------+----------+-------+-------+---------------------+---------------------+--------+
  2 rows in set (0.02 sec)
  ```

- **fs.obs.access.key**: AK information obtained in **Step 9**
- **fs.obs.secret.key**: SK information obtained in **Step 9**
- **fs.obs.endpoint**: endpoint information obtained in **Step 5**

Step 4 Check the task progress.

show load order by createtime desc limit 1\G;

If the value of **State** in the command output changes to **FINISHED**, the data import is complete.

```
JobId: 296805
Label: brokerload_test_csv_label001
State: FINISHED
Progress: ETL:100%; LOAD:100%
Type: BROKER
EtlInfo: unselected.rows=0; dpp.abnorm.ALL=0; dpp.norm.ALL=1000000
TaskInfo: cluster:N/A; timeout(s):14400; max_filter_ratio:0.0
 ErrorMsg: NULL
...
```

Step 5 Query table data.

select * from *test*;

----End

5.5.3 Importing Data to Doris with Stream Load

Stream Load is a synchronous import mode. Users send requests through HTTP to import local files or data streams to the Doris. Stream Load synchronously executes the import and returns the import result. You can determine whether the import is successful based on the return body of the request.

Stream Load is used to import local files or import data in data flows through programs. Data in CSV, Parquet, and ORC formats can be imported. Only data in CSV format is supported by default.

Syntax

- Creating a Stream Load Import Task

  Stream load submits and transfers data through HTTP protocol. This operation uses the curl command to demonstrate how to submit the import. You can also use other HTTP clients to perform this operation.

  – Kerberos authentication is enabled for the cluster (the cluster is in security mode)

    curl -k --location-trusted -u *user:passwd* **[-H ""...] -T data.file -XPUT https://***Doris FE instance IP address:HTTPS port number*/api/*{Database name}*/*{Table name}*/**_stream_load**

  – Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

    curl --location-trusted -u *user:passwd* **[-H ""...] -T data.file -XPUT http://***Doris FE instance IP address:HTTP port number*/api/*{Database name}*/*{Table name}*/**_stream_load**

  To view the IP address of the Doris FE instance, log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris** > **Instances**.

  You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and search for **https_port** to view the HTTPS port.

  . To view the port, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and search for **http_port**.

  Table 5-2 describes other parameters for creating a stream load task.

Table 5-2 Parameters of the Stream Load Task

Parameter		Description
Signature parameters	user:passwd	The HTTP protocol is used for stream load creation and import, and the basic access authentication is used for signature. The Doris system verifies user identity and import permissions based on signatures.

Parameter		Description
Import task parameters (format: **-H "key1:value1"**)	label	Identity of import task. Each import task has a unique label inside a single database. Label is a user-defined name in the import command. Through this label, you can view the execution status of the corresponding import task.
	column_separator	Specifies the column separator in the file to be imported. The default value is **\t**. You can use a combination of multiple characters as the column separator. If it is an invisible character, you need to add **\x** as a prefix and hexadecimal to indicate the separator. For example, the separator **\x01** of the hive file needs to be specified as **-H "column_separator:\x01"**.
	line_delimiter	Specifies the newline character in the file to be imported. The default value is **\n**. You can use to combine multiple characters as the newline character.
	max_filter_ratio	The maximum tolerance rate of the import task is 0 by default, and the range of values is 0-1. When the import error rate exceeds this value, the import fails.
	where	Filter criteria specified for an import task. Stream Load allows you to specify **where** statements to filter raw data. The filtered data will not be imported or involved in **filter ratio** calculation, but will be counted in **num_rows_unselected**.
	Partitions	Partition information of the table to be imported. If the data to be imported does not belong to the specified partition, the data will not be imported and will be counted in dpp.abnorm.ALL.
	columns	Function transformation configuration of the data to be imported. Currently, the function transformation methods supported by Stream Load include column sequence transformation and expression transformation. The expression transformation method is the same as that of the query statement.
	format	Format of the data to be imported. The value can be CSV, JSON, Parquet, or ORC. The default value is CSV.
	exec_mem_limit	Memory limit for data import. The default value is 2 GB. The unit is byte.

Parameter		Description
	strict_mode	The strict mode can be enabled for stream load import. You can declare **strict_mode=true** in the header to enable the strict mode. By default, the strict mode is disabled. The strict mode is used to strictly filter column type conversion during data import. The policy is as follows: ● For column type conversion, if **strict mode** is true, incorrect data will be filtered. Error data refers to the data whose original data is not null and whose result is null after column type conversion. ● If a column to be imported is converted by a function, the strict mode does not affect the column. ● If the type of a column to be imported contains a range restriction and the original data can be converted but cannot pass the range restriction, the strict mode does not affect the column. For example, if the type is decimal (1,0) and the original data is 10, the data can be converted but is not within the range specified by the column. The strict data does not affect the data.
	merge_type	Data combination type. The options are APPEND, DELETE, and MERGE. The default value is APPEND. ● APPEND indicates that the batch of data needs to be added to the existing data. ● DELETE indicates that all rows that have the same key as the batch of data are deleted. ● MERGE semantics must be used together with the **delete** condition. Data that meets the delete condition is processed based on DELETE semantics, and other data is processed based on APPEND semantics.
	two_phase_commit	The two-phase transaction commit mode can be enabled for stream load import. During stream loading, a message is returned to you after data is written. In this case, the data is invisible and the transaction status is PRECOMMITTED. The data is visible only after you manually trigger the **commit** operation.
	enable_profile	If enable_profile is set to true, the stream load profile is recorded in logs. Otherwise, the stream load profile is not recorded in logs.

- Returned Result

  Stream Load is a synchronous import mode. Therefore, the import result is directly returned through the return value of the import creation. For example:

  ```
  {
  "TxnId": 1003,
  ```

```
"Label": "b6f3bc78-0d2c-45d9-9e4c-faa0a0149bee",
"Status": "Success",
"ExistingJobStatus": "FINISHED", // optional
"Message": "OK",
"NumberTotalRows": 1000000,
"NumberLoadedRows": 1000000,
"NumberFilteredRows": 1,
"NumberUnselectedRows": 0,
"LoadBytes": 40888898,
"LoadTimeMs": 2144,
"BeginTxnTimeMs": 1,
"StreamLoadPutTimeMs": 2,
"ReadDataTimeMs": 325,
"WriteDataTimeMs": 1933,
"CommitAndPublishTimeMs": 106,
"ErrorURL": "http://192.168.1.1:8042/api/_load_error_log?file=__shard_0/
error_log_insert_stmt_db18266d4d9b4ee5-
abb00ddd64bdf005_db18266d4d9b4ee5_abb00ddd64bdf005"
}
```


Table 5-3 describes the parameters in the import result.

Table 5-3 Parameters in the Stream Load Import Task Result

Parameter	Description
TxnId	The imported transaction ID.
Label	Import labels, which are specified by users or automatically generated by the system.
Status	Import completion status. The options are as follows: • Success: indicates that the import is successful. • Publish Timeout: This status also indicates that the import is complete. The only difference is that the data may be visible after a delay and does not need to be retried. • Label Already Exists: The label already exists and needs to be replaced. • Fail: Import failed.
ExistingJobStatus	Status of the import job corresponding to an existing label. This field is displayed only when Status is **"Label Already Exists"**. You can view the status of the import job corresponding to the existing label. **"RUNNING"** indicates that the job is still being executed, and **"FINISHED"** indicates that the job is successfully executed.
Message	Import error information.
NumberTotalRows	Number of rows imported for total processing.
NumberLoadedRows	Number of rows successfully imported.
NumberFilteredRows	Number of rows in which data does not meet requirements.

Parameter	Description
NumberUnselectedRows	Number of rows filtered by the **where** condition.
LoadBytes	Number of imported bytes
LoadTimeMs	Import completion time, in milliseconds.
BeginTxnTimeMs	This parameter specifies the time taken to request the FE to start a transaction. The unit is millisecond.
StreamLoadPutTimeMs	Time taken to obtain the data import plan from the FE. The unit is millisecond.
ReadDataTimeMs	Time taken to read data, in milliseconds.
WriteDataTimeMs	Time taken to write data, in milliseconds.
CommitAndPublishTimeMs	Time taken to submit a request to the FE and release a transaction, in milliseconds.
ErrorURL	If a data error occurs, access the URL to view the error line.

NOTE

Stream Load is imported in synchronous mode. Therefore, the import information is not recorded in the Doris system. Users cannot view Stream Load asynchronously by running the import command. You need to view the return value of the import request to obtain the import result.

- Cancel Import

  You cannot manually cancel stream load tasks. Stream load will be automatically canceled by the system after a timeout or import error.

- Viewing Stream Load

  You can view completed stream load tasks on the **show stream load**.

  By default, the BE does not record the stream load import information. To view the information, you need to set the **enable_stream_load_record=true** parameter.

Prerequisite

- A cluster containing the Doris service has been created, and all services in the cluster are running properly.
- The node to be connected to the Doris database can communicate with the MRS cluster.
- A user with Doris management permission has been created.
  - Kerberos authentication is enabled for the cluster (the cluster is in security mode)

Log in to FusionInsight Manager, create a human-machine user, for example, **dorisuser**, create a role with Doris administrator permissions, and bind the role to the user.

Log in to FusionInsight Manager as the created **dorisuser** user, and change the initial password.

- Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

  After connecting to Doris as user **admin**, create a role with administrator permissions, and bind the role to the user.

- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.

Stream Load Task Example

Step 1 Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login user* **-p***Database login user password* **-P***Database connection port* **-h***Doris FE instance IP address*

NOTE

- The database connection port is the query connection port of the Doris FE. You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris WebUI to connect to the database.

Step 2 Run the following command to create a database:

create database if not exists *example_db*;

Step 3 Run the following statement to create a table:

CREATE TABLE *example_db.test_stream_tbl* **(**

`c1` int NOT NULL,

`c2` int NOT NULL,

`c3` string NOT NULL,

`c4` date NOT NULL

) ENGINE=OLAP

UNIQUE KEY(`c1`, `c2`)

DISTRIBUTED BY HASH(`c1`) BUCKETS 1;

Step 4 Create the data.csv file. The file content is as follows:

```
1,1,1,2020-02-21
2,2,2,2020-03-21
3,3,3,2020-04-21
```

Step 5 Use Stream Load to import data in the **data.csv** file to the table created in **Step 3**.

- Kerberos authentication is enabled for the cluster (the cluster is in security mode)

  curl -k --location-trusted -u *user:passwd* -H "*label:table1_20230217***" -H "***column_separator:,***" -T data.csv https://***IP address of the Doris FE instance:HTTPS port***/api/***example_db***/***test_stream_tbl***/_stream_load**

- Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

  curl --location-trusted -u *user:passwd* -H "*label:table1_20230217***" -H "***column_separator:,***" -T data.csv http://***IP address of the Doris FE instance:HTTP port***/api/***example_db***/***test_stream_tbl***/_stream_load**

Step 6 Run the following command to view the table data:

select * from *example_db.test_stream_tbl*;

----End

Related Parameter Configurations

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations** > **All Configurations**, and add the following parameters:

- Choose **FE(Role)** > **Customization**, and add the following parameters to the customized parameter **fe.conf.customized.configs**:

  stream_load_default_timeout_second: timeout interval of an import task, in seconds. If an import task is not complete within the specified time, the system cancels the task and the task status changes to CANCELLED. The default timeout period is 600 seconds. If the source file to be imported cannot be imported within the specified period, you can set a separate timeout period in the Stream Load request or change the value of **stream_load_default_timeout_second** to set the global default timeout period.

- Choose **BE(Role)** > **Customization** and add the following parameters to the custom parameter **be.conf.customized.configs**:

  streaming_load_max_mb: maximum size of a file to be imported for Stream Load. The default value is 10 GB, in MB. If the size of the original file exceeds the value of this parameter, you need to adjust the value of this parameter.

5.6 Analyzing Doris Data

5.6.1 Exporting Doris Data to HDFS

The data export function can export data in specified tables or partitions to remote storage devices in text format through the Broker process, such as HDFS and object storage (supporting the S3 protocol).

NOTE

- You are not advised to export a large amount of data at a time. It is recommended that a maximum of dozens of GB data be exported for an export job. Large exports result in more junk files and higher retry costs.
- If the table contains a large amount of data, you are advised to export the data by partition.
- If the FE is restarted or an active/standby switchover occurs during the running of the Export job, the Export job fails and you need to submit the job again.
- If the export job fails, the temporary directory **__doris_export_tmp_***xxx* generated in the remote storage and the generated files will not be deleted. You need to manually delete them.
- If the export job is successfully executed, the **__doris_export_tmp_***xxx* directory generated in the remote storage may be retained or cleared based on the file system semantics of the remote storage.

  For example, in object storage (supporting the S3 protocol), after the last file in a directory is moved through the **rename** operation, the directory is also deleted. If the directory is not cleared, you can manually clear it.
- After the export is complete (successful or failed), if the FE is restarted or an active/standby switchover occurs, some job information displayed in **SHOW EXPORT** will be lost and cannot be viewed.
- The Export job exports only Base table data and does not export Rollup Index data.
- The Export job scans data and occupies I/O resources, which may affect the query delay of the system.

Syntax

- Exporting Doris Data to HDFS
  - Kerberos authentication is enabled for the cluster (the cluster is in security mode)

    EXPORT TABLE *db1.tbl1*

    PARTITION (p1,p2)

    [WHERE [expr]]

    TO "hdfs://*IP address of the active NameNode instance*:*RPC port number*/**tmp/export/"**

    PROPERTIES

    (

    "label" = "*mylabel***",**

    "column_separator"=",",

    "columns" = "col1,col2",

    "exec_mem_limit"="2147483648",

    "timeout" = "3600"

)

    WITH BROKER "broker_name"

    (

    "hadoop.security.authentication"="kerberos",

    "kerberos_principal"="doris/hadoop.hadoop.com@HADOOP.COM",

    "kerberos_keytab"="${BIGDATA_HOME}/FusionInsight_Doris_*/install/FusionInsight-Doris-*/doris-fe/bin/doris.keytab"

);

- Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

  EXPORT TABLE *db1.tbl1*

  PARTITION (p1,p2)

  [WHERE [expr]]

  TO "hdfs://*IP address of the active NameNode instance***:***RPC port number***/tmp/export/"**

  PROPERTIES

  (

  "label" = "*mylabel***",**

  "column_separator"=",",

  "columns" = "col1,col2",

  "exec_mem_limit"="2147483648",

  "timeout" = "3600"

)

  WITH BROKER "broker_name"

  (

  "username" = "*user***",**

  "password" = "*passwd***"**

  );

To view the IP address of the active NameNode instance, log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS** > **Instances**.

You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** > **Configurations**, and search for **dfs.namenode.rpc.port** to view the RPC port.

Table 5-4 describes other parameters.

Table 5-4 Parameters in the command for exporting Doris data to HDFS

Parameter	Description
label	URN of the export job. You can use this identifier to view the job status.
column_separator	Column separator. The default value is **\t**. Invisible characters are supported, for example, **'\x07'**.
columns	Columns to be exported. Use commas (,) to separate them. If this parameter is not set, all columns in the table are exported by default.
line_delimiter	Line separator. The default value is **\n**. Invisible characters are supported, for example, **'\x07'**.
exec_mem_limit	Memory usage limit of a query plan on a BE in an export job. The default value is 2 GB, in bytes.

Parameter	Description
timeout	Job timeout interval. The default value is 2 hours. The unit is second.
tablet_num_per_task	Maximum number of shards allocated to each query plan. The default value is 5.

- Viewing the Status of an Export Job

  After submitting a job, you can run the **SHOW EXPORT;** command to query the status of the export job. For example:

```
JobId: 14008
State: FINISHED
Progress: 100%
TaskInfo: {"partitions":["*"],"exec mem limit":2147483648,"column separator":",","line
delimiter":"\n","tablet num":1,"broker":"hdfs","coord num":1,"db":"default_cluster:db1","tbl":"tbl3"}
Path: hdfs://host/path/to/export/
CreateTime: 2019-06-25 17:08:24
StartTime: 2019-06-25 17:08:28
FinishTime: 2019-06-25 17:08:34
Timeout: 3600
ErrorMsg: NULL
1 row in set (0.01 sec)
```

  Table 5-5 describes the parameters.

  Table 5-5 Parameters for exporting job status

Parameter	Description
JobId	Job ID, which is unique.
State	Job status. The options are as follows: • PENDING: The job is to be scheduled. • EXPORTING: Data is being exported. • FINISHED: The job is successfully exported. • ANCELLED: The export job fails to be executed.
Progress	Job progress, in the unit of query plan. Assume that there are 10 query plans in total and three of them have been completed. The progress is 30%.

Parameter	Description
TaskInfo	Job information displayed in JSON format, where: • db: database name. • tbl: table name • partitions: specifies the partition to be exported. * indicates all partitions. • exec mem limit: queries the memory usage limit of the plan. in byte. • column separator: column separator of the exported file. • line delimiter: line delimiter of the exported file. • tablet num: total number of tablets. • broker: name of the used broker. • coord num: number of query plans.
Path	Export path on the remote storage device.
CreateTime/ StartTime/ FinishTime:	Creation time, scheduling start time, and end time of a job.
Timeout	Job timeout interval, in seconds. The time starts from CreateTime.
ErrorMsg	If an error occurs in the job, ErrorMsg displays the cause of the error.

- Canceling an Export Task

  After submitting a job, you can run the **CANCEL_EXPORT** command to cancel the export job. The cancellation command is as follows:

  CANCEL EXPORT

  FROM *example_db*

  WHERE LABEL like "*%example%***";**

Prerequisite

- A cluster containing the Doris service has been created, and all services in the cluster are running properly.
- The node to be connected to the Doris database can communicate with the MRS cluster.
- A user with Doris management permission has been created.
  - Kerberos authentication is enabled for the cluster (the cluster is in security mode)

    Log in to FusionInsight Manager, create a human-machine user, for example, **dorisuser**, create a role with Doris administrator permissions, and bind the role to the user.

    Log in to FusionInsight Manager as the created **dorisuser** user, and change the initial password.

  - Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

    After connecting to Doris as user **admin**, create a role with administrator permissions, and bind the role to the user.

- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.

Example of Exporting Doris Data to HDFS

Step 1 Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login user* **-p***Database login user password* **-P***Database connection port* **-h***Doris FE instance IP address*

NOTE

- The database connection port is the query connection port of the Doris FE. You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris WebUI to connect to the database.

Step 2 Run the following command to create a database:

create database if not exists *example_db*;

Step 3 Run the following command to create a table:

CREATE TABLE *example_db.test_export_tbl* **(**

`c1` int NOT NULL,

`c2` int NOT NULL,

`c3` string NOT NULL,

`c4` date NOT NULL

) ENGINE=OLAP

DUPLICATE KEY(`c1`, `c2`)

DISTRIBUTED BY HASH(`c1`) BUCKETS 1;

Step 4 Run the following command to insert data:

insert into example_db.test_export_tbl values(1,1,1,"2020-02-21"), (2,2,2,"2020-03-21"),(3,3,3,"2020-04-21");

Step 5 Run the following command to export data from the test_export_tbl table to HDFS:

EXPORT TABLE *example_db.test_export_tbl*

TO "hdfs:// IP address of the active NameNode instance:RPC port number */tmp/export/***"**

PROPERTIES

(

"label" = "*label_exporthdfs_20230218031***",**

"column_separator"=",",

"columns" = "c1,c2,c3,c4",

"exec_mem_limit"="2147483648",

"timeout" = "3600"

)

with broker "broker_192_168_67_78"

(

"hadoop.security.authentication"="kerberos",

"kerberos_principal"="doris/hadoop.hadoop.com@HADOOP.COM",

"kerberos_keytab"="${BIGDATA_HOME}/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-fe/bin/doris.keytab"

);

Step 6 Run the following command to query the status of the export job:

SHOW EXPORT;

----End

Related Configuration Parameters

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations** > **FE(Role)** > **Customization**, and add the following parameters to the customized parameter fe.conf.customized.configs:

- export_checker_interval_second: scheduling interval of the Export job scheduler. The default value is 5 seconds. After setting this parameter, restart the FE.
- export_running_job_num_limit: specifies the maximum number of running export jobs. If the value is exceeded, the job waits and is in the **PENDING** state. The default value is 5, which can be adjusted during running.
- export_task_default_timeout_second: default timeout interval of an export job. The default value is 2 hours, which can be adjusted during running.
- export_tablet_num_per_task: maximum number of shards that a query plan is responsible for. The default value is 5.
- label: label of an EXPORT task manually specified by a user. If this parameter is not specified, a label is automatically generated.

5.6.2 Exporting the Query Result Set

This section describes how to use the **SELECT INTO OUTFILE** command to export the Doris query result set to a specified storage system in a specified file format.

NOTE

- The export command does not check whether the file and file path exist. The semantics of the remote storage system determines whether to automatically create a path or overwrite an existing file.
- If an error occurs during the export, some exported files may remain on the remote storage system. Doris does not clear these files. You need to manually clear them.
- The timeout period of the export command is the same as that of the query command. You can set the timeout period through **SET query_timeout=***xxx*.
- For a query whose result set is empty, a file whose size is 0 is still generated.
- File splitting ensures that a row of data is completely stored in a single file. Therefore, the file size is not strictly equal to the value of max_file_size.
- For some functions whose output is invisible characters, such as BITMAP and HLL, the output is **\N**, that is, NULL.
- Currently, the output type of some geographic information functions, such as ST_Point, is VARCHAR. However, the actual output value is encoded binary characters. These functions output garbled characters. For geographic functions, use ST_AsText for output.

Syntax

query_stmt

INTO OUTFILE "file_path"

[format_as]

[properties]

file_path

format_as

properties

NOTE

format_as: indicates the export format. The value can be CSV, PARQUET, CSV_WITH_NAMES, CSV_WITH_NAMES_AND_TYPES or ORC. The default value is CSV.

Example

- Export to HDFS

  Export the simple query result to the hdfs://path/to/result.txt file in CSV format.

  - Kerberos authentication is enabled for the cluster (the cluster is in security mode)

    SELECT * FROM example_db.test_export_tbl

    INTO OUTFILE "hdfs://*192.168.67.78:25000***/tmp/result_"**

    FORMAT AS CSV

    PROPERTIES

(

"broker.name" = "broker_192_168_67_78",

"column_separator" = ",",

"line_delimiter" = "\n",

"max_file_size" = "100MB",

"broker.hadoop.security.authentication" = "kerberos",

"broker.kerberos_principal" = "doris/hadoop.hadoop.com@HADOOP.COM",

"broker.kerberos_keytab" = "${BIGDATA_HOME}/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-fe/bin/doris.keytab"

);

– Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

SELECT * FROM example_db.test_export_tbl

INTO OUTFILE "hdfs://*192.168.67.78:25000***/tmp/result_"**

FORMAT AS CSV

PROPERTIES

(

"broker.name" = "broker_192_168_67_78",

"column_separator" = ",",

"line_delimiter" = "\n",

"max_file_size" = "100MB",

"broker.username"="hdfs",

"broker.password"=""

);

- Export to Local File

  Before exporting data to a local file, you need to configure **enable_outfile_to_local=true in the fe.conf file**.

  select * from *tbl1* **limit 10**

  INTO OUTFILE "file:///*home/work/path/result_***";**

5.7 Enterprise-Class Enhancements of Doris

5.7.1 Configuring HA for the Doris Cluster

5.7.1.1 About Doris Cluster HA

Clients that support the MySQL protocol can be connected to the Doris cluster through FE nodes. To avoid single points of failure (SPOFs), multiple FE nodes are used and ELB instances are deployed on these nodes to ensure high availability of Doris.

You can configure Doris HA use any of the following methods base on your needs:

- Write code on the service side.
- Use an SDK.
- Configure an ELB instance.

Writing Code on the Service Side

At the service application layer, write code to test the connection and balance loads. If a connection is interrupted, the connection is automatically set up on other FEs for retry. To retry the connection, you need to make intrusive modification of addresses of multiple Doris FE nodes.

Using an SDK

The MySQL protocol is used to connect to Doris. SDKs of some languages have provided high availability. For example, MySQL JDBC can use the automatic retry mechanism. When establishing a connection, set the data source as follows:

jdbc:mysql:loadbalance://[host1][:port],[host2][:port][,[host3][:port]]...[/[database]][?
propertyName1=propertyValue1[&propertyName2=propertyValue2]...]

For details, see **https://dev.mysql.com/doc/connector-j/en/connector-j-usagenotes-j2ee-concepts-managing-load-balanced-connections.html**.

Configuring an ELB Instance

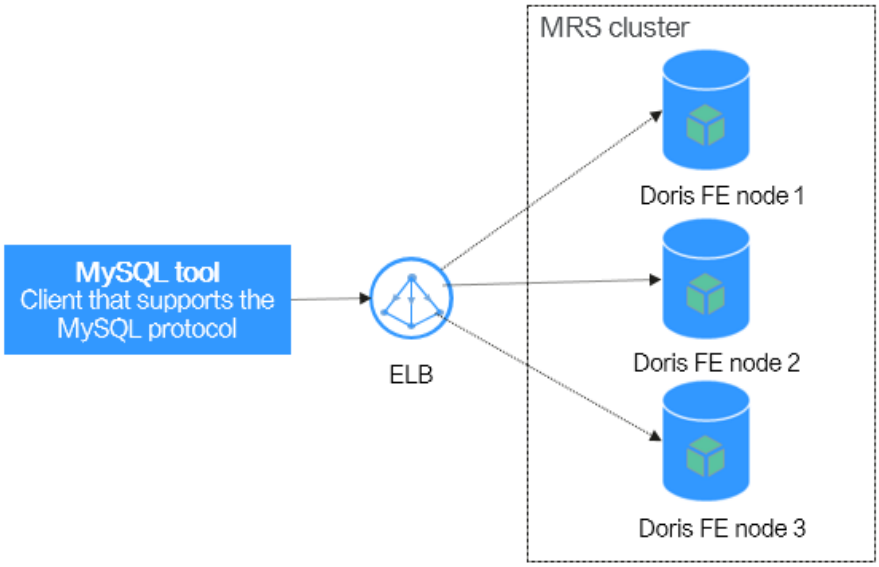
Doris uses the ELB-based deployment architecture to automatically distribute user access traffic to multiple backend nodes, expanding service capabilities to external systems and improving fault tolerance. When a backend Doris node becomes faulty, ELB automatically fails over access traffic to another properly running node. For details, see **Configuring Access to the Doris Cluster Through ELB**.

5.7.1.2 Configuring Access to the Doris Cluster Through ELB

Doris can use a client that supports the MySQL protocol to access a single FE node to perform service operations. If FE nodes are faulty, Doris cannot provide services for external systems. To solve this problem, MRS can be deployed with the Elastic Load Balance (ELB) service in the architecture shown in **Figure 5-1**.

Doris uses the ELB-based deployment architecture to automatically distribute user access traffic to multiple backend nodes, expanding service capabilities to external systems and improving fault tolerance. When a backend Doris node becomes faulty, ELB automatically fails over access traffic to another properly running node.

Figure 5-1 Accessing Doris nodes through ELB



Enable the MySQL client to access Doris through ELB. The procedure is as follows:

- **Step 1: Purchase an ELB and obtain its public IP address.**
- **Step 2: Add an ELB listener and configure its protocol and port.**
- **Step 3: Access Doris through ELB on the MySQL client.**

Prerequisite

- A Doris cluster has been created and is running properly.
- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.

Purchasing ELB and Interconnecting It with Doris

Buy an ELB instance and obtain its public IP address.

For details, see **Creating a Shared Load Balancer**.

Step 1 Log in to the management console and choose **Networking** > **Elastic Load Balance (ELB)** from the service list.

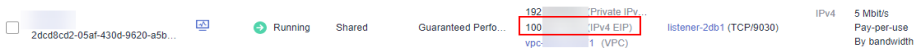
Step 2 In the upper right corner of the ELB console, click **Buy Elastic Load Balancer**.

Step 3 On the displayed page, set the following parameters and retain the default values for other parameters:

- **Type**: **Shared load balancer**
- **Billing Mode**: **Pay-per-use**
- **Enterprise Project**: **default**
- The values of **VPC** and **Frontend Subnet** must be the same as those of the MRS Doris cluster.

Step 4 Click **Next**, confirm the configurations, and click **Submit**.

Step 5 On the **Load Balancers** page, view and obtain the public IP address of the newly created load balancer.

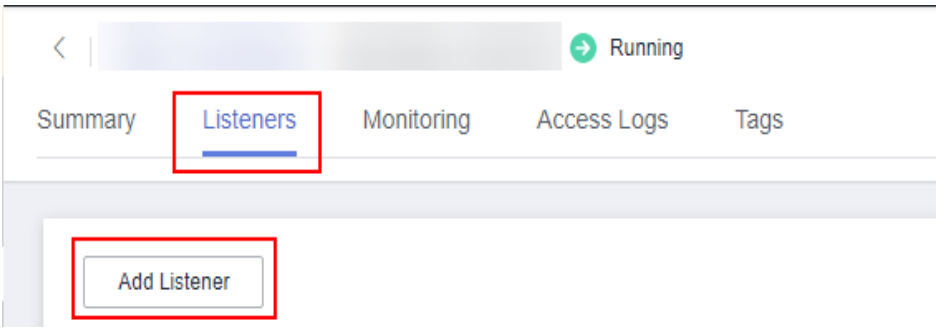


Add an ELB listener.

For details, see **Adding a Listener**.

Step 6 On the **Load Balancers** page, click the name of the created load balancer to go to its details page.

Step 7 Click the **Listeners** tab and then **Add Listener**.



Step 8 On the **Add Listener** page, set the parameters as prompted. Retain the default values for the parameters that are not mentioned in the following steps.

1. Configure the listener.

   Set **Frontend Protocol** to **TCP**, set **Frontend Port** to the access port, and click **Next: Configure Request Routing Policy**.

2. Configure a routing policy.

   Set **Load Balancing Algorithm** to **Weighted round robin**, enable **Sticky Session**, and click **Next: Add Backend Server**.

3. Add backend servers.

   In the **Backend Servers** pane, click **Add Backend Server**, add all nodes where Doris FE is deployed, and click **OK**.

   NOTE

   To view the service IP address of a Doris FE instance, log in to the MRS cluster management console, choose **Components**, click **Doris**, and click the **Instances** tab.

4. Set **Backend Port** of the servers to the MySQL protocol query connection port of the Doris FE service. The default port is 9030. You can search for **query_port** on the service configuration page of the Doris component to view the port.
5. Click **Next: Confirm**.
6. Confirm the configuration and click **Submit**.
7. Click **View/Add Backend Server** in the row where the created listener is located. In the **Backend Servers** pane, check whether the connection between the ELB and backend server is normal.

Use ELB to access Doris on the MySQL client.

Step 9 Log in to a node where MySQL is installed and run the following command to connect to Doris. For details, see **Getting Started with Doris**.

mysql -u *Database login user* **-p***Password* **-P***ELB frontend port* **-h***ELB public IP address*

NOTE

- The ELB frontend port is that configured in **8.1**.
- Replace the ELB public IP address with the IP address obtained in **Step 5**.

Step 10 Run the following command to check the connection status of the FE service:

show frontends;

If the query result is successful, Doris can be successfully accessed through ELB.

----End

5.7.2 Configuring Multi-Source Data for Doris

5.7.2.1 About Doris Multi-Source Data

The multi-source data catalog aims to facilitate interconnection with external data catalogs to enhance Doris' data lake analysis and federated data query capabilities.

The multi-source data catalog function adds a catalog layer to the original metadata layer to form the three metadata layers of the Catalog -> Database -> Table. The catalog may directly correspond to the external data catalog.

Basic Concepts

- Internal Catalog

  The original databases and tables of the Doris belong to the Internal Catalog. Internal Catalog is a built-in default catalog and cannot be modified or deleted by users.

- External Catalog

  You can run the **CREATE CATALOG** command to create an External Catalog, and view the existing Catalogs using the **SHOW CATALOGS** command.

- Switching Catalogs

  After login, you will enter the Internal Catalog by default (the default usage is the same as that in earlier versions). Then, you can view or switch to your target database via **SHOW DATABASES** and **USE DB**.

  You can run the SWITCH command to switch the catalog. For example:

  ```
  SWITCH internal;
  SWITCH hive_catalog;
  ```

  After the switchover, you can run the **SHOW DATABASES** and **USE DB** commands to view and switch the database in the corresponding catalog. The Doris automatically synchronizes the databases and tables in the catalog. You can view and access data in External Catalogs the same way as doing that in Internal Catalogs.

Currently, the Doris supports only read-only access to data in the External Catalog.

- Delete Catalog

  Databases and tables in External Catalog are read-only. However, the catalog can be deleted (the internal catalog cannot be deleted). You can run the **DROP CATALOG** command to delete an External Catalog.

  This operation only deletes the mapping information of the Catalog in Doris, but does not modify or change the content of any external data catalog.

- Resource

  Resource is a set of configurations. You can run the CREATE RESOURCE command to create a resource. Then, you can use the resource when creating a catalog.

  A resource can be used by multiple catalogs to reuse the configuration in the resource.

5.7.2.2 Interconnecting Doris with the Hive Data Source

By connecting to Hive Metastore, or a metadata service compatible with Hive Metatore, Doris can automatically obtain Hive database table information and perform data queries.

In addition to Hive, many other systems also use Hive Metastore to store metadata. Through Hive Catalog, you can access not only Hive but also systems that use Hive Metastore as metadata storage, such as Iceberg and Hudi.

NOTE

- Managed tables are supported.
- Hive and Hudi metadata stored in Hive Metastore can be identified.
- If you want to access the catalog that is not created by the current user, you need to grant the user the permission to operate the OBS path where the catalog resides.
- The Hive table format can only be Parquet, ORC, or TextFile.

Prerequisite

- A cluster containing the Doris service has been created, and all services in the cluster are running properly.
- The node to be connected to the Doris database can communicate with the MRS cluster.
- A user with Doris management permission has been created.
  - Kerberos authentication is enabled for the cluster (the cluster is in security mode)

    Log in to FusionInsight Manager, create a human-machine user, for example, **dorisuser**, create a role with Doris administrator permissions, and bind the role to the user.

    Log in to FusionInsight Manager as the created **dorisuser** user, and change the initial password.
  - Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

After connecting to Doris as user **admin**, create a role with administrator permissions, and bind the role to the user.

- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.
- If Doris imports data across clusters through Broker Load, you need to configure cross-cluster mutual trust. For details, see **Configuring Cross-Manager Mutual Trust Between Clusters**.

Hive Table Operations

Step 1 To use Doris to read Hive data stored in OBS, perform the following operations:

1. Log in to the Huawei Cloud management console. On the console page, move the cursor to the username in the upper right corner and select **My Credentials** from the drop-down list.
2. Click **Access Keys**, click **Create Access Key**, and enter the verification code or password. Click **OK** to generate an access key, and download it.

   Obtain the values of AWS_ACCESS_KEY and AWS_SECRET_KEY required for creating a catalog from the .csv file. The mapping is as follows:

   - The value of AWS_ACCESS_KEY is the value in the Access Key Id column in the .csv file.
   - The value of AWS_SECRET_KEY is the value in the Secret Access Key column of the .csv file.

   NOTE

   - Keep the CSV file properly. You can only download the file right after the access key is created. If you cannot find the file, you can create an access key again.
   - Keep your access keys secure and change them periodically for security purposes.
   - In MRS 3.3.1 and later versions, **AWS_ACCESS_KEY** and **AWS_SECRET_KEY** are replaced with **obs.access_key** and **obs.secret_key**.
3. You can obtain the value of **AWS_REGION** (**obs.region** in MRS 3.3.1 and later versions) from **Regions and Endpoints**.
4. Log in to the OBS management console, click **Parallel File System**, click the name of the OBS parallel file system where the Hive table resides, and view the value of **Endpoint** on the overview page. The value is the same as that of **AWS_ENDPOINT** (**obs.endpoint** in MRS 3.3.1 and later versions) set during catalog creation.

Step 2 Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login username* **-p***Database login password* **-P***Database connection port* **-h***IP address of the Doris FE instance*

NOTE

- The database connection port is the query connection port of the Doris FE. You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris WebUI to connect to the database.

Step 3 Create a catalog.

- Hive table data is stored in HDFS. Run the following command to create a catalog:
  - Kerberos authentication is enabled for the cluster (the cluster is in security mode):

    CREATE CATALOG hive_catalog PROPERTIES (

    'type'='hms',

    'hive.metastore.uris' = 'thrift://*192.168.67.161:21088***',**

    'hive.metastore.sasl.enabled' = 'true',

    'hive.server2.thrift.sasl.qop' = 'auth-conf',

    'hive.server2.authentication' = 'KERBEROS',

    'dfs.nameservices'='hacluster',

    'dfs.ha.namenodes.hacluster'='*24,25***',**

    'dfs.namenode.rpc-address.hacluster.*24***='** IP address of the active NameNode:RPC communication port **',**

    'dfs.namenode.rpc-address.hacluster.*25***='** IP address of the standby NameNode:RPC communication port **',**

    'dfs.client.failover.proxy.provider.hacluster'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider',

    'hive.version' = '3.1.0',

    'yarn.resourcemanager.address' = '*192.168.67.78:26004***',**

    'yarn.resourcemanager.principal' = 'mapred/hadoop.hadoop.com@HADOOP.COM',

    'hive.metastore.kerberos.principal' = 'hive/hadoop.hadoop.com@HADOOP.COM',

    'hadoop.security.authentication' = 'kerberos',

    'hadoop.kerberos.keytab' = '${BIGDATA_HOME}/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-be/bin/doris.keytab',

    'hadoop.kerberos.principal' = 'doris/hadoop.hadoop.com@HADOOP.COM',

    'java.security.krb5.conf' = '${BIGDATA_HOME}/FusionInsight_BASE_*/1_16_KerberosClient/etc/krb5.conf',

    'hadoop.rpc.protection' = 'privacy'

    );

- Kerberos authentication is disabled for the cluster (the cluster is in normal mode):

  CREATE CATALOG hive_catalog PROPERTIES (

  'type'='hms',

  'hive.metastore.uris' = 'thrift://*192.168.67.161:21088***',**

  'hive.version' = '3.1.0',

  'hadoop.username' = 'hive',

  'yarn.resourcemanager.address' = '*192.168.67.78:26004***',**

  'dfs.nameservices'='hacluster',

  'dfs.ha.namenodes.hacluster'='24,25',

  'dfs.namenode.rpc-address.hacluster.24'='192-168-67-172:25000',

  'dfs.namenode.rpc-address.hacluster.25'='192-168-67-78:25000',

  'dfs.client.failover.proxy.provider.hacluster'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'

  );

NOTE

- hive.metastore.uris: URL of Hive MetaStore. The format is **thrift://***<Hive MetaStore IP address>***:***<Port number>*. Multiple values are supported and separated by commas (,).
- dfs.nameservices: NameService name of the cluster. The value can be found in **hdfs-site.xml**, which is in the **${BIGDATA_HOME}/FusionInsight_HD_*/1_*_NameNode/etc** directory where NameNode is deployed.
- dfs.ha.namenodes.hacluster: cluster NameService prefix, which contains two values. The value can be found in **hdfs-site.xml**, which is in the **${BIGDATA_HOME}/FusionInsight_HD_*/1_*_NameNode/etc** directory where NameNode is deployed.
- dfs.namenode.rpc-address.hacluster.*xx1*: specifies the RPC communication address of the active NameNode. You can search for the value of this configuration item in **hdfs-site.xml** in the **${BIGDATA_HOME}/FusionInsight_HD_*/1_*_NameNode/etc** directory on the node where NameNode resides. *xx* is the value of **dfs.ha.namenodes.hacluster**.
- dfs.namenode.rpc-address.hacluster.*xx2*: RPC communication address of the standby NameNode. You can search for the value of this configuration item in **hdfs-site.xml** in the **${BIGDATA_HOME}/FusionInsight_HD_*/1_*_NameNode/etc** directory on the node where NameNode resides. *xx* is the value of **dfs.ha.namenodes.hacluster**.
- To query the IP address of the active/standby instance, choose **Cluster** > **Services** > **HDFS** > **Instances** on FusionInsight Manager.
- The default RPC port number is **25000**. To query it, log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** > **Configurations**.
- dfs.client.failover.proxy.provider.hacluster: specifies the Java class for the HDFS client to connect to the active node in the cluster. The value is **org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider**.
- hive.version: Hive version. To obtain the version, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Hive**, and view the version on the **Dashboard** page.
- yarn.resourcemanager.address: indicates the IP address of the active ResourceManager instance. On FusionInsight Manager, choose **Cluster** > **Services** > **Yarn** > **Instances** to view the service IP address of the active ResourceManager instance. To obtain the port number, click **Configurations** and search for **yarn.resourcemanager.port** in the search box.
- hadoop.rpc.protection: specifies whether to encrypt the RPC stream of each Hadoop module. The default value is privacy. To obtain the value, log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** > **Configurations**, and search for **hadoop.rpc.protection**.
- Kerberos authentication is enabled for the cluster (the cluster is in security mode):
  - hive.metastore.sasl.enabled: indicates whether to enable the MetaStore management permission. The value is **true**.
  - hive.server2.thrift.sasl.qop: indicates whether to encrypt the interaction between HiveServer2 and the client. The value is **auth-conf**.
  - hive.server2.authentication: security authentication mode for accessing HiveServer. The value is **KERBEROS**.
  - yarn.resourcemanager.principal: Principal for accessing the Yarn cluster. The value is **mapred/hadoop.hadoop.com@HADOOP.COM**.
  - hive.metastore.kerberos.principal: Principal for accessing the Hive cluster. The value is **hive/hadoop.hadoop.com@HADOOP.COM**.
  - hadoop.security.authentication: security authentication mode for accessing Hadoop. The value is **KERBEROS**.

    - hadoop.kerberos.keytab: keytab for accessing the Hadoop cluster. The value is the path of the **${BIGDATA_HOME}/FusionInsight_Doris_*/install/FusionInsight-Doris-*/doris-be/bin/doris.keytab** file.
    - hadoop.kerberos.principal: Principal for accessing the Hadoop cluster. The value is **doris/hadoop.hadoop.com@HADOOP.COM**.
    - java.security.krb5.conf: krb5 file. The value is the path of the **${BIGDATA_HOME}/FusionInsight_BASE_*/1_*_KerberosClient/etc/krb5.conf** file.
  - Kerberos authentication is disabled for the cluster (the cluster is in normal mode):

    hadoop.username: user name for accessing the Hadoop cluster. The value is **hdfs**.

- Hive table data is stored in OBS. Run the following command to create a catalog. For details about related parameter values, see **Step 1**.

  CREATE CATALOG *hive_obs_catalog* **PROPERTIES (**

  'type'='hms',

  'hive.version' = '3.1.0',

  'hive.metastore.uris' = 'thrift://*192.168.67.161:21088***',**

  'hive.metastore.sasl.enabled' = 'true',

  'hive.server2.thrift.sasl.qop' = 'auth-conf',

  'hive.server2.authentication' = 'KERBEROS',

  'dfs.nameservices'='hacluster',

  'dfs.ha.namenodes.hacluster'='*24***,***25***',**

  'dfs.namenode.rpc-address.hacluster.*24***='** *IP address of the active NameNode:RPC communication port* **',**

  'dfs.namenode.rpc-address.hacluster.*25***='** *IP address of the standby NameNode:RPC communication port* **',**

  'dfs.client.failover.proxy.provider.hacluster'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider',

  'yarn.resourcemanager.address' = '*192.168.67.78:26004***',**

  'yarn.resourcemanager.principal' = 'mapred/hadoop.hadoop.com@HADOOP.COM',

  'hive.metastore.kerberos.principal' = 'hive/hadoop.hadoop.com@HADOOP.COM',

  'hadoop.security.authentication' = 'kerberos',

  'hadoop.kerberos.keytab' = '${BIGDATA_HOME}/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-be/bin/doris.keytab',

  'hadoop.kerberos.principal' = 'doris/hadoop.hadoop.com@HADOOP.COM',

  'java.security.krb5.conf' = '${BIGDATA_HOME}/FusionInsight_BASE_*/1_16_KerberosClient/etc/krb5.conf',

  'AWS_ACCESS_KEY' = '*AK***',**

  'AWS_SECRET_KEY' = '*SK***',**

  'AWS_ENDPOINT' = ' Endpoint address of the OBS parallel file system **',**

  'AWS_REGION' = '*sa-fb-1***',**

  'hadoop.rpc.protection' = 'privacy'

  );

Step 4 Run the following command to query the Hive table:

- Run the following command to query Catalog:
  show catalogs;
- Run the following command to query the database in the Catalog:
  show databases from *hive_catalog*;
- Run the following command to switch to the Catalog and access the database:
  switch *hive_catalog*;
  use *default*;
- To query all tables in a database in Catalog, run the following command:
  show tables from **`***hive_catalog***`.`***default***`;**
  To query a specified table, run the following command:
  select * from **`***hive_catalog***`.`***default***`.`***test_table***`;**
  Run the following command to view the schema of the table:
  DESC *test_table*;

Step 5 After creating or operating a Hive table, you need to refresh the table in the Doris.

refresh catalog *hive_catalog*;

Step 6 Run the following command to perform associated query with tables in other data directories:

SELECT h.h_shipdate FROM *hive_catalog.default.htable* **h WHERE h.h_partkey IN (SELECT p_partkey FROM internal.db1.part) LIMIT 10;**

NOTE

- A table is identified in **catalog.database.table** full restriction mode, for example, internal.db1.part.
- **catalog** and **database** can be omitted. By default, the Catalog and Database after the switchover of **SWITCH** and **USE** are used.
- You can run the **INSERT INTO** command to insert table data in Hive Catalog to an internal table in Interal Catalog to import external data catalog data.

----End

5.7.2.3 Interconnecting Doris with the Hudi Data Source

When MRS cluster metadata is stored in Hive MetaStore, Doris 2.0.13 can connect to data sources of Hudi 0.15.0 using Catalog and supports all data field types of Hudi.

Hudi Table Types Can Be Queried by Doris

The supported Hudi table types and corresponding query types are as follows:

- COW table: Snapshot Query and TimeTravel Query
- MOR table: Snapshot Query, TimeTravel Query, and Read Optimized Query

The following uses TimeTravel Query as an example to describe how to query a Hudi COW table.

Step 1 Log in to the node where MySQL is installed and connect the Doris database.

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login user* **-p -P***Connection port for FE queries* **-h***IP address of the Doris FE instance*

Enter the password for logging in to the database.

NOTE

- To obtain the query connection port of the Doris FE instance, you can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris web UI to connect the database.

Step 2 Create a Hudi Catalog, for example, **hudi_catalog**. For details about how to create, see **Step 3**.

Step 3 Connect the MySQL client to Doris by referring to **Step 1** and run the following command to switch to the Hudi Catalog created in **Step 2**:

switch *hudi_catalog*;

Step 4 Run the following command to view the created Hudi table:

refresh catalog *hudi_catalog*;

use *Database name*;

show tables;

Step 5 Use TimeTravel Query to view the table data inserted before a certain time:

select * from *hudi_cow* **for time as of** '*20240409162842365*' **where id =** *1*;

----End

Doris on Hudi Query Acceleration

Doris on Hudi supports the following query acceleration:

- Bucket Shuffle Joins on Hudi Parquet Table

  Bucket Shuffle Join is available in Doris to accelerate the queries on Hudi data sources. This operation is supported for all Hudi field types and Hudi tables stored on OBS. The session-level variable **enable_hudi_bucket_shuffle** controls whether to enable this function, which is defaulted to disabled. You can connect to Doris and run the **set enable_hudi_bucket_shuffle=true** command to enable this function.

NOTE

- The inner join condition contains bucketing columns of two tables. When the bucketing columns of the left table are inner join conditions, there is a high probability that bucket shuffle join is planned.
- The type of the bucketing column in the left table must be the same as that of the inner join column in the right table.
- The bucket shuffle join function is available only in inner joins.
- The bucket shuffle join function is available only when the left table has a single partition.
- Two tables to be joined, one can be a Hudi table, and the other can be a Doris internal table.

- Hudi Implicit Partitioning (Restricted Feature)

  Doris supports Hudi's implicit partitioning for query acceleration. You can use **the enable_hudi_hidden_partition** parameter to enable or disable this function (which is disabled by default). To enable it, run the **ADMIN SET FRONTEND CONFIG ("enable_hudi_hidden_partition" = "true");** command after connecting to Doris.

- Hudi Table Bucket Pruning (Restricted Feature)

  Doris supports Hudi bucket pruning for faster queries. You can use **enable_hudi_bucket_prunning** parameter to enable or disable this function (which is disabled by default). To enable it, run the **ADMIN SET FRONTEND CONFIG ("enable_hudi_bucket_prunning" = "true");** command after connecting to Doris.

Step 1 Log in to the node where MySQL is installed and connect the Doris database.

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login user* **-p -P***Connection port for FE queries* **-h***IP address of the Doris FE instance*

Enter the password for logging in to the database.

NOTE

- To obtain the query connection port of the Doris FE instance, you can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris web UI to connect the database.

Step 2 Create a Hudi Catalog, for example, **hudi_catalog**. For details about how to create, see **Step 3**.

Step 3 Connect to Doris on the MySQL client by referring to **Step 1** and run the following command to switch to Hudi Catalog:

switch *hudi_catalog*;

Step 4 Run the following commands to view the created Hudi table:

refresh catalog *hudi_catalog*;

use *Database name*;

show tables;

Step 5 Enable Doris on Hudi query acceleration and perform a query.

- Enable Bucket Shuffle Join and perform a query.

  Only Hudi bucket index tables are supported. That is, the Hudi table to be queried has the **hoodie.index.type='BUCKET'**, **hoodie.bucket.index.num.buckets='***xxx***'** and **hoodie.bucket.index.hash.field='***xxx***'** properties.

  set enable_hudi_bucket_shuffle = true;

  select t1.name, t2.age from *hudi_testt1* **t1,** *hudi_testt2* **t2 where t1.id=t2.id;**

  NOTE

  Check whether the Bucket Shuffle Join is hit.

  explain select t1.name, t2.age from *hudi_testt1* **t1,** *hudi_testt2* **t2 where t1.id=t2.id;**

  Check whether the keyword **BUCKET_SHUFFLE** exists in the command output. If it exists, the query hits the Bucket Shuffle Join.

- Enable implicit partitioning and query a Hudi table.

  Only implicitly partitioned Hudi tables are supported. That is, Hudi table to be queried has the **hoodie.hidden.partitioning.rule = '***xxx***'** and **hoodie.hidden.partitioning.enabled = 'true'** properties.

  ADMIN SET FRONTEND CONFIG ("enable_hudi_hidden_partition" = "true");

  select * from hudi1 where col0 like '%2';

- Enable the bucket pruning function and query a Hudi table.

  Only Hudi simple bucket index tables are supported. That is, the Hudi table to be queried has the **hoodie.index.type = 'BUCKET'**, **hoodie.bucket.index.num.buckets = '***xxx***'**, **hoodie.bucket.index.hash.field = '***xxx***'**, **hoodie.index.bucket.engine = 'SIMPLE'**, and **hoodie.metadata.index.column.stats.enable = 'true'** properties.

  ADMIN SET FRONTEND CONFIG ("enable_hudi_bucket_prunning" = "true");

  select * from hudi_cow_tbl where id > 1 and id < 3;

----End

5.7.2.4 Configuring Spark to Read and Write Doris Data

The Spark Doris Connector can read data stored in the Doris through Spark and write data to the Doris through Spark.

- Data can be read from the Doris.
- Spark DataFrame can be written to Doris in batches or in streaming mode.
- You can map a Doris table to a DataFrame or RDD. DataFrame is recommended.

- Data can be filtered at the Doris end to reduce the amount of data to be transferred.
- The operations in this section are applicable to MRS 3.5.0 and later versions.

Prerequisite

- A cluster containing the Doris service has been created, and all services in the cluster are running properly.
- The node to be connected to the Doris database can communicate with the MRS cluster.
- A user with Doris management permission has been created.
  - Kerberos authentication is enabled for the cluster (the cluster is in security mode)

    Log in to FusionInsight Manager, create a human-machine user, for example, **dorisuser**, create a role with Doris administrator permissions, and bind the role to the user.

    Log in to FusionInsight Manager as the created **dorisuser** user, and change the initial password.
  - Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

    After connecting to Doris as user **admin**, create a role with administrator permissions, and bind the role to the user.
- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.
- The Spark client has been installed.

Procedure

Create a table in Doris and insert data into the table.

Step 1 Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login username* **-p***Database login password* **-P***Database connection port* **-h***IP address of the Doris FE instance*

NOTE

- The database connection port is the query connection port of the Doris FE. You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris WebUI to connect to the database.

Step 2 Run the following commands to create a database and switch the database:

create database if not exists *sparkconnector*;

use *sparkconnector*;

Step 3 Run the following statement to create a table:

CREATE TABLE *spark_connector_test_decimal* **(**

c1 int NOT NULL,

c2 VARCHAR(25) NOT NULL,

c3 VARCHAR(152),

c4 boolean,

c5 tinyint,

c6 smallint,

c7 bigint,

c8 float,

c9 double,

c10 date,

c11 datetime,

c12 char,

c13 largeint,

c14 varchar,

c15 decimal(15, 5)

)

DUPLICATE KEY(`c1`)

COMMENT "OLAP"

DISTRIBUTED BY HASH(`c1`) BUCKETS 1;

Step 4 Run the following commands to insert data to the table:

insert into *spark_connector_test_decimal* **values(10000,'aaa','abc',true, 100, 3000, 100000, 1234.567, 12345.678, '2022-12-01','2022-12-01 12:00:00', 'a', 200000, 'g', 1000.12345);**

insert into *spark_connector_test_decimal* **values(10001,'aaa','abc',false, 100, 3000, 100000, 1234.567, 12345.678, '2022-12-01','2022-12-01 12:00:00', 'a', 200000, 'g', 1000.12345);**

insert into *spark_connector_test_decimal* **values(10002,'aaa','abc',True, 100, 3000, 100000, 1234.567, 12345.678, '2022-12-01','2022-12-01 12:00:00', 'a', 200000, 'g', 1000.12345);**

insert into *spark_connector_test_decimal* **values(10003,'aaa','abc',False, 100, 3000, 100000, 1234.567, 12345.678, '2022-12-01','2022-12-01 12:00:00', 'a', 200000, 'g', 1000.12345);**

Perform operations on Spark.

Step 5 Run the following command to log in to the spark-sql client:

cd Spark client installation directory

source bigdata_env

kinit *Component service user* (skip this step if Kerberos authentication is disabled for the cluster (the cluster is in normal mode))

spark-sql --master yarn

Step 6 Run the following command to create a temporary view:

CREATE TEMPORARY VIEW *spark_doris_decimal*

USING doris

OPTIONS(

"table.identifier"="sparkconnector.spark_connector_test_decimal",

"fenodes=" FE instance IP address **:***Port number***",**

"user"="*dorisuser***",**

"password"="*User password***",**

'doris.enable.https' = 'true',

'doris.ignore.https.ca' = 'true'

);

Run the following command to query the data in the Doris table:

select * from *spark_doris_decimal***;**

Run the following command to insert data into the Doris table:

insert into *spark_doris_decimal* **values(10005,'aaa','abc',False, 100, 3000, 100000, 1234.567, 12345.678, '2022-12-01','2022-12-01 12:00:00', 'a', 200000, 'g', 1000.12345);**

NOTE

- After switching to HTTP, delete the following configuration parameters from the WITH clause for creating a table:
  - **'doris.enable.https' = 'true'**
  - **'doris.ignore.https.ca' = 'true'**
- The port number is the HTTPS port of the FE service (for clusters with Kerberos authentication enabled) or HTTP port (for clusters with Kerberos authentication disabled). You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and enter **https_port** or **http_port** in the search box.
- When the amount of data to be imported is too large, you can adjust the following parameters to improve the performance:
  - **sink.batch.size**: maximum number of rows that can be written to a BE at a time. The default value is **10000**.
  - **doris.sink.batch.interval.ms**: indicates the interval between each batch of sinks, in milliseconds. The default value is **50**.

----End

5.7.2.5 Configuring Flink to Read and Write Doris Data

Flink Doris Connector allows you to perform operations (read, insert, modify, and delete) on data stored in Doris through Flink.

NOTE

- Only tables of the Unique Key model can be modified or deleted.
- The operations in this section are applicable to MRS 3.5.0 and later versions.

Prerequisite

- A cluster containing the Doris service has been created, and all services in the cluster are running properly.
- The node to be connected to the Doris database can communicate with the MRS cluster.
- A user with Doris management permission has been created.
  - Kerberos authentication is enabled for the cluster (the cluster is in security mode)

    Log in to FusionInsight Manager, create a human-machine user, for example, **dorisuser**, create a role with Doris administrator permissions, and bind the role to the user.

    Log in to FusionInsight Manager as the created **dorisuser** user, and change the initial password.
  - Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

    After connecting to Doris as user **admin**, create a role with administrator permissions, and bind the role to the user.
- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.
- The Flink client has been installed.

Procedure

Doris side operation.

Step 1 Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login username* **-p***Database login password* **-P***Database connection port* **-h***IP address of the Doris FE instance*

NOTE

- The database connection port is the query connection port of the Doris FE. You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris WebUI to connect to the database.

Step 2 Run the following commands to create a database and switch the database:

create database if not exists *testdb***;**

use *testdb***;**

Step 3 Run the following commands to create the **z_test** table and insert data into the table:

create table *z_test***(id int, name string) distributed by hash(id) buckets 10;**

insert into *z_test* **values(123, 'aaa'), (234, 'bbb'), (345, 'ccc');**

Step 4 Run the following command to create the **z_test_sink_3** table:

create *table z_test_sink_3***(id int, name string) distributed by hash(id) buckets 10;**

Perform operations on Flink.

Step 5 Log in to the node where the Flink client is installed as the client installation user and run the following command:

cd *Client installation directory*

source bigdata_env

kinit *Component service user* (If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), skip this step.)

Step 6 Run the following command to log in to the Flink SQL client:

cd Flink/flink/bin/

sql-client.sh

Step 7 Create a stream or batch Flink streaming SQL job on the Flink client. The following command is an example:

CREATE TABLE *flink_doris_source* **(id INT, name STRING) WITH (**

'connector' = 'doris',

'fenodes' ='FE instance IP address:*Port number*',

'table.identifier' = 'testdb.z_test',

'username' = '*user***',**

'password' = '*password***',**

'doris.enable.https' = 'true',

'doris.ignore.https.ca' = 'true'

);

CREATE TABLE *flink_doris_sink* **(id INT, name STRING) WITH (**

'connector' = 'doris',

'fenodes' = ' FE instance IP address **:***Port number***',**

'table.identifier' = 'testdb.z_test_sink_3',

'username' = '*user***',**

'password' = '*password***',**

'sink.label-prefix' = 'doris_label_6',

'doris.enable.https' = 'true',

'doris.ignore.https.ca' = 'true'

);

Run the following commands to insert data:

INSERT INTO

flink_doris_sink

select

id,

name

from

*flink_doris_source***;**

NOTE

- After HTTPS is enabled, add the following configuration parameters to the **with** clause for creating a table:
  - **'doris.enable.https' = 'true'**
  - **'doris.ignore.https.ca' = 'true'**
- The fields in the source and sink tables must be the same as those in the Doris table.
- The port number is the HTTPS port of the FE service (for clusters with Kerberos authentication enabled) or HTTP port (for clusters with Kerberos authentication disabled). You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and enter **https_port** or **http_port** in the search box.
- When you create a Flink job, set **username** to the Doris user and **password** to the password of the Doris user.
- If Kerberos authentication (security mode) has been enabled for the cluster, only the HTTPS mode can be configured.

----End

5.7.2.6 Connecting to the MySQL/Doris Data Source Through JDBC Catalog

Scenario

JDBC Catalog can connect to data sources such as MySQL and Doris through the standard JDBC protocol. This topic describes how to use JDBC Catalog to read MySQL or Doris data.

The operations in this section are applicable to MRS 3.3.1 and later versions.

Prerequisites

- An MRS cluster has been created and the Doris service has been deployed.
- A source cluster, for example, a MySQL server cluster, has been prepared.
- The Doris cluster can communicate with the MySQL server cluster.
- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.

Querying MySQL/Doris Data Through JDBC Catalog

Step 1 Visit the MySQL official website at **https://www.mysql.com/**, choose **DOWNLOADS** > **MySQL Community(GPL) DownLoads** > **Connector/J**, and download the driver package of the required version.

NOTE

The recommended MySQL driver version is 5.1.*x* or later.

Step 2 Upload the downloaded MySQL driver package to the same directory on all FE and BE nodes, for example, **/home/omm**, and change the owner group of the driver package to **omm:wheel**.

NOTE

If Doris FE and BE are deployed on the same nodes, upload the MySQL driver package to the **${BIGDATA_HOME}/FusionInsight_Doris_*/install/FusionInsight-Doris-*/doris-fe/jdbc_drivers/** directory.

Step 3 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris** > **Configurations** > **All Configurations** > **FE(Role)** > **Customization**.

Step 4 Add the **jdbc_drivers_dir** parameter to the custom parameter **fe.conf.customized.configs**. The value is the directory where the driver package uploaded to in **Step 2**, for example, **/home/omm**.

Step 5 Choose **All Configurations** > **BE(Role)** > **Customization**. Add the **jdbc_drivers_dir** parameter to the **fe.conf.customized.configs** parameter and set it to the directory where the driver package uploaded to in **Step 2**, for example, **/home/omm**.

Step 6 Search for the **enable_udf_sandbox** parameter of BE and set it to **false**.

Step 7 Click **Save**. On the **Dashboard** page, click **More** > **Restart Service** in the upper right corner. Enter the password of the user and click **OK** to restart the Doris service.

Step 8 Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login user* **-p -P***Connection port for FE queries* **-h***IP address of the Doris FE instance*

Enter the password for logging in to the database.

NOTE

- To obtain the query connection port of the Doris FE instance, you can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the service IP address of any FE instance.
- You can also use the MySQL connection software or Doris web UI to connect to the database.

Step 9 Create a JDBC catalog in the target Doris cluster.

- If the source cluster is a MySQL server and Kerberos authentication is disabled for the target Doris cluster, run the following commands:

  CREATE CATALOG *doris_jdbc_catalog* **PROPERTIES (**

  "type"="jdbc",

  "user"=" *MySQL username* **",**

  "password"="*MySQL user password***",**

  "jdbc_url" = "jdbc:mysql://*IP address of MySQL*:*MySQL port* **?useSSL = false",**

  "driver_url" = "mysql-connector-java-*xxx***.jar",**

  "driver_class" = "com.mysql.jdbc.Driver"

  );

- If both the source and destination are MRS Doris clusters and Kerberos authentication is disabled for the clusters, run the following commands:

  CREATE CATALOG *doris_jdbc_catalog* **PROPERTIES (**

  "type"="jdbc",

  "user"=" *Doris username* **",**

  "password"="*Doris user password***",**

  "jdbc_url" = "jdbc:mysql:// *IP address of the source Doris FE instance*:*FE query connection port* **?useSSL = false",**

  "driver_url" = "mysql-connector-java-*xxx***.jar",**

  "driver_class" = "com.mysql.jdbc.Driver"

  );

- If both the source and destination are MRS Doris clusters and Kerberos authentication is enabled for the clusters, run the following commands:

  CREATE CATALOG *doris_jdbc_catalog* **PROPERTIES (**

  "type"="jdbc",

  "user"=" *Doris username* **",**

  "password"="*Doris user password***",**

  "jdbc_url" = "jdbc:mysql://*IP address of the source Doris FE instance*:*FE query connection port***?allowPublicKeyRetrieval=true&useSSL=true&verifyServerCertificate=false"**

  ,

  "driver_url" = "mysql-connector-java-*xxx***.jar",**

  "driver_class" = "com.mysql.jdbc.Driver"

  );

NOTE

- If the MySQL driver version is 6.x or later, add the **serverTimezone=GMT%2B8** parameter to **jdbc_url** when creating a JDBC catalog to prevent creation failure caused by time zone inconsistency between the source cluster and the current Doris cluster.
- If the MySQL driver version is 6.x or later, the value of **driver_class** is **com.mysql.cj.jdbc.Driver**. If the MySQL driver version is 5.x, the value of **drive_class** is **com.mysql.jdbc.Driver**.
- To obtain the query connection port of the Doris FE instance, you can log in to Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance on the source cluster, log in to the Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.

Step 10 Run the following commands to switch to the newly created catalog and view the database:

switch doris_jdbc_catalog;

show databases;

If the MySQL database name and table name are displayed in the command output, Doris can access the MySQL cluster through JDBC Catalog.

----End

5.7.3 Configuring Doris Multi-Tenancy

5.7.3.1 Overview

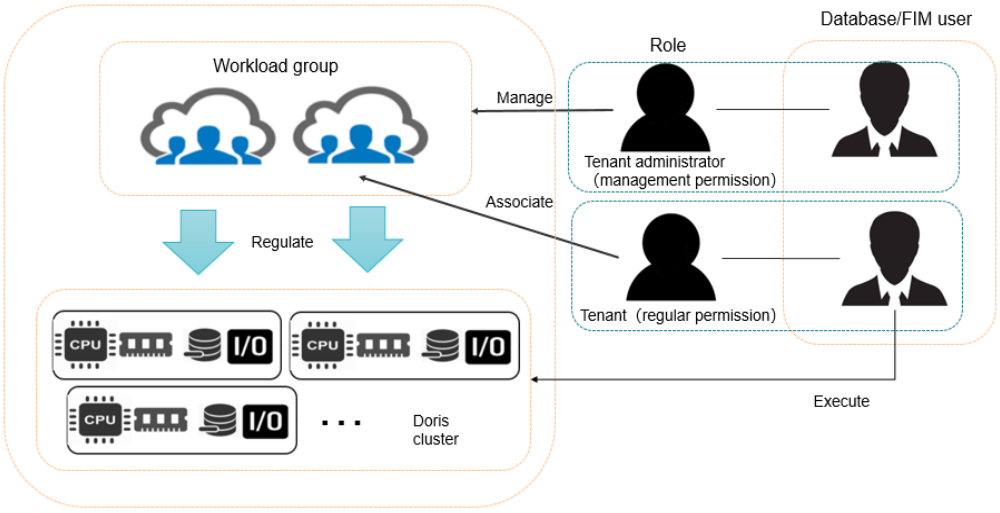
Doris Multi-Tenancy

Doris multi-tenancy is built on top of the Workload Group Resource soft limit of the kernel. Workloads are managed by group to ensure flexible allocation and control of memory and CPU resources. Each tenant role of a user can have many workload groups. You can manage CPU, memory, concurrency, and queues with this model shown in **Figure 5-2**.

NOTE

- (This feature is available for MRS 3.3.1 or later only.)
- Multiple tenants can be created and managed only on FusionInsight Manager when Kerberos authentication is enabled for the cluster (the cluster is in security mode).
- In MRS 3.5.0 and later versions: The default memory usage of the **normal** tenant is 90%, and the number of concurrent requests is 10,000.

Figure 5-2 Doris tenant model

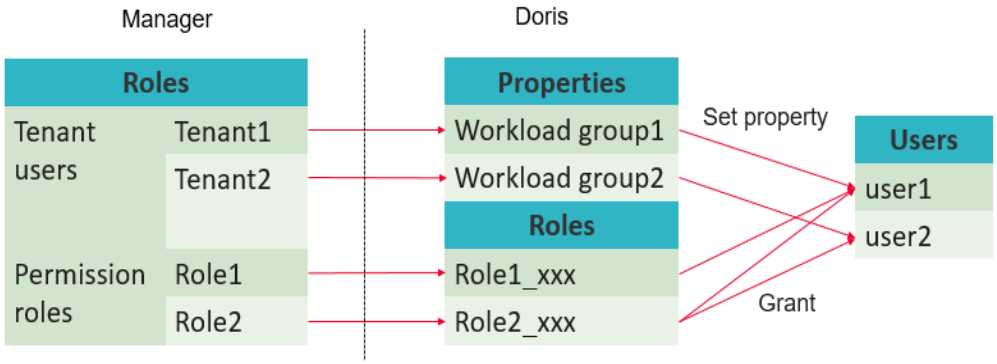


After a user is associated with a tenant, all query tasks submitted by the user are added to the workload group. You can limit the percentage of CPU and memory resources on BE nodes for a single query and configure a soft memory limit for the workload group.

When cluster resources are insufficient, the system automatically stops several query tasks that occupy memory the most in the group. If the resources used by a workload group exceed the preset limit, multiple workloads share idle resources in the cluster and use resources more than the limit. This ensures stable execution of query tasks. In addition, the query queuing function is introduced to relieve system pressure in heavy-load scenarios. When creating a workload group, you can set the maximum number of concurrent queries to queue the exceeding queries.

Associations Between Doris Tenant Roles and Users

On the FusionInsight Manager service configuration and tenant management pages, you can create tenants, associate tenants with services, configure tenant resources, and associate tenants with users. The following figure shows the association between roles and users on the Manager and Doris sides.



A user, a tenant role, and a workload group are in one-on-one relationship. A user can have multiple permission roles.

Table 5-6 lists the Doris tenant resource configurations supported by the current version.

Table 5-6 tenant resource configuration

Configuration	Value Range	Description	Remarks
CPU Quota Usage	1 to 100	Weight of CPU resources that can be used by a tenant	The value specifies a relative ratio that is valid during resource competition. For example, if this value for tenant A is 10 and that for tenant B is 20, the CPU resources can be used by the query tasks of tenant A is one-third of total resources, that is, 10/(10 + 20). If tenant C starts query tasks and its CPU quota usage is 30, a CPU quota of tenant A is one-sixth of total resources, that is, 10/(10 + 20 + 30).

Configuration	Value Range	Description	Remarks
Memory Quota	• MRS 3.3.1: 1%~70% • MRS 3.5.0 and later versions: 1%~90%	Maximum proportion of memory that can be used by a tenant	The available memory of a tenant is calculated as follows: Physical memory x **mem_limit** x Memory quota. The upper limit of the memory quota is 70% (90% for MRS 3.5.0 and later versions). The default Doris tenant **normal** occupies 70% (90% for MRS 3.5.0 and later versions) of memory.
Concurrency	• MRS 3.3.1: 1~2147483647 • MRS 3.5.0 and later versions: 1~10000	Maximum number of concurrent query tasks that a tenant can run	This parameter specifies the maximum number of tasks on each FE node. For example, if the number of concurrent SQL statements is set to 1 and the Doris has three FE nodes, the maximum number of SQL statements that can be executed in a cluster is 3.
Queue Length	• MRS 3.3.1: 0~2147483647 • MRS 3.5.0 and later versions: 0~1000	Maximum number of waiting query tasks	Excessive SQL statements are queued. When the queue is full, newly submitted queries are rejected.
Waiting Duration	• MRS 3.3.1: 0~2147483647 • MRS 3.5.0 and later versions: 0~86400000	Maximum waiting duration of a tenant query task	If the query waiting duration exceeds the value of this parameter, the query is rejected. The unit is millisecond.

Configuration	Value Range	Description	Remarks
Soft Memory Limit	• On • Off	Whether a tenant can use more memory resources than the limit	• If this function is disabled, the system immediately cancels the tasks that occupy the most memory in the tenant groups when detecting that the memory usage of the tenant exceeds the upper limit. • If this function is enabled and the cluster has idle memory resources, the tenant can use the system memory more than the limit. The tasks that occupy the most memory in the tenant groups are canceled only when cluster resources are insufficient.

5.7.3.2 Managing Doris Tenants

The cluster administrator can create a Doris tenant on FusionInsight Manager.

Creating a Doris Tenant

Step 1 Log in to FusionInsight Manager and choose **Tenant Resources**.

Step 2 On the **Tenant Resources Management** page, click . On the displayed page, configure tenant properties by referring to **Table 5-7**.

Table 5-7 Tenant parameters

Parameter	Description
Name	Specify the name of the current tenant. The value can contain 3 to 50 characters but cannot contain only digits. Only digits, letters, and underscores (_) are allowed. Plan a tenant name based on service requirements. The name cannot be the same as that of a role, HDFS directory, or YARN queue that exists in the current cluster.
Tenant Type	Select **Leaf Tenant**. **NOTE** A Doris tenant can only be a leaf tenant.

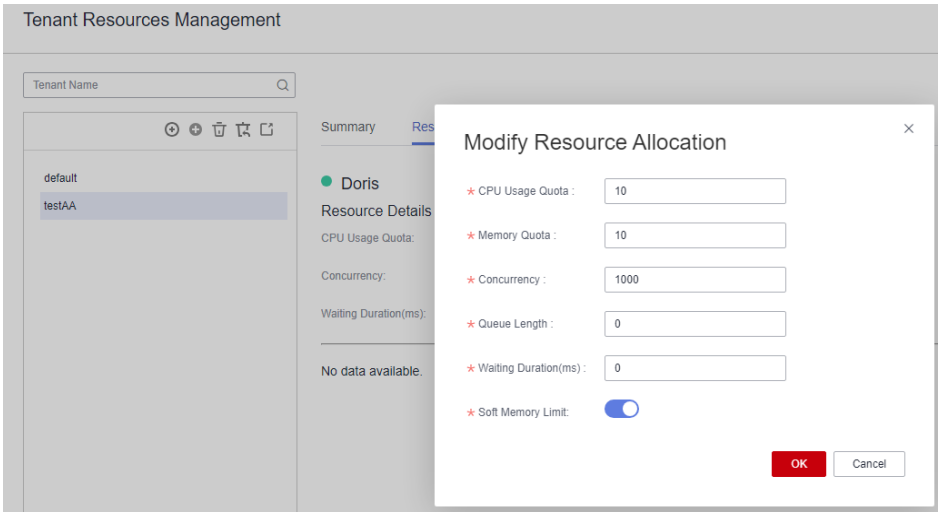
Parameter	Description
Compute Resource	Do not select **Yarn** if only Doris-related tenants are created.
Storage Resource	Do not select **HDFS** if only Doris-related tenants are created.
Service	Click **Associate Service**. In the displayed dialog box, set the following parameters and click **OK**: • **Service**: Select **Doris**. • **Association Type**: Maintain the default option **Shared**. For details about the following parameters, see **Table 5-6**. • **CPU Usage Quota**: specifies the weight of CPU resources occupied by a tenant. This parameter is valid during resource competition. • **Memory Quota**: specifies the maximum percentage of memory resources occupied by a tenant. For example, if this parameter is set to 10, the available memory of the tenant on each BE node is 10% of the available memory on the BE node. The default Doris tenant **normal** occupies 90% of the resources. Therefore, for other tenants this parameter can be set up to 10%. If the sum exceeds 100%, the Doris tenant fails to be created. • **Concurrency**: specifies the maximum number of concurrent query tasks a tenant can run. Excessive query tasks are queued. • **Queue Length**: specifies the maximum number of query tasks waiting in the queue. • **Waiting Duration**: specifies the maximum waiting duration of a query task. The unit is milliseconds. • **Soft Memory Limit**: By default, soft memory limit is disabled. If this option is enabled and when resources are sufficient, query tasks can use resources more than the limit. When resources are insufficient, the occupied memory is released. If this function is disabled and when the memory limit of a workload group is reached, som SQL statements will be canceled or rejected.
Description	Description of the tenant.

Step 3 Click **OK**. Wait until the tenant is created.

Step 4 View and modify the tenant in the **Tenant Resources Management** page.

1. On FusionInsight Manager, click **Tenant Resources**. In the tenant list, select the desired Doris tenant and view the tenant information and the resource quota.

2. Click **Resource** and click next to **Resource Details** to modify tenant resources.



3. Click **OK**. The details are displayed in the **Resource** tab.

NOTE

- In the **Summary** tab, resource quota is not updated in real time. It is updated only when it is loaded.
- A Doris tenant represents a workload group, which limits the compute resources of tasks in the group on a single instance node. The **Resource Quota** and **Chart** information show monitoring data of the average metric values. The chart is refreshed every 30 seconds.

----End

Associating an Existing Tenant with Doris and a User

Step 1 On FusionInsight Manager, choose **Tenant Resources**. In the tenant list, select the desired tenant, click the **Service Associations** tab, and click **Associate Service**.

Step 2 On the displayed dialog box, set **Service** to **Doris**, set other parameters as you need, and click **OK** to associate the tenant with the Doris service.

Step 3 Click the **User** tab, and then click **Associate User** on the displayed page. Associate the tenant with a user, and click **OK**.

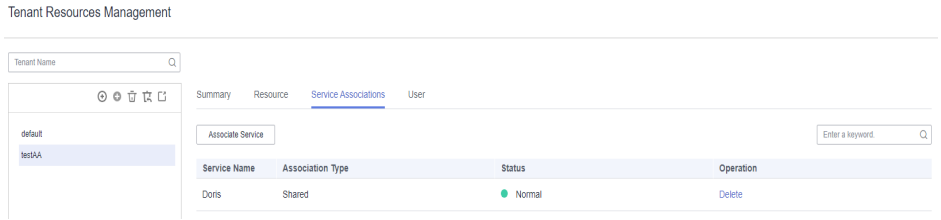
NOTE

- You can also create a user or associate a tenant with an existing user on the user list page by referring to **Adding a User and Binding It to a Tenant**.
- A tenant can be associated with multiple users.
- Doris tenants can be associated with human-machine and machine-machine users created on Manager.
  - If Kerberos authentication is enabled for the cluster (the cluster is in security mode), associate the tenant with a human-machine user.
  - If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), associate the tenant with a machine-machine user.
- If a common Doris user is associated (that is, the role of the user does not have the Doris administrator permission), the following error is reported when the user connects to Doris on the MySQL client, creates a table, and queries the table data:
  ERROR 1227 (42000): errCode = 2, detailMessage = Access denied; you need (at least one of) the USAGE/ADMIN privilege(s) to use workload group '*Tenant name*'.

  Connect to Doris on the MySQL client as a user with the Doris administrator permission and run the following command to grant permission to the current user:

  GRANT USAGE_PRIV ON workload group '*Tenant name* **' to '***User ame* **';**

Step 4 To disassociate the Doris service, click **Delete** in the **Operation** column of the Doris service. In the displayed dialog box, click **OK**.



NOTE

After a tenant is disassociated from the Doris service, the workload group of the Doris kernel is deleted, and the workload group of the users bound to the tenant is also set to **normal**.

----End

Adding a User and Binding It to a Tenant

- Create a user and bind it to a tenant: Log in to FusionInsight Manager, choose **System** > **Permission** > **User**, click **Create**, add a human-machine user (a machine-machine user if Kerberos authentication is disabled for the cluster (the cluster is in normal mode)), and add the created Doris tenant to the role.
- Bind a tenant to an existing user: Log in to FusionInsight Manager, choose **System** > **Permission** > **User**, click **Modify** in the **Operation** column of the user, and add the created Doris tenant to the role.

NOTE

- To delete a Doris tenant, choose **System** > **User**, locate the row that contains the user in the user list, click **Modify**, and delete the Doris tenant bound added to the role.
- A user cannot be bound to multiple Doris tenants. For example, if **user1** has been associated with **tenant1** of **Doris**, unbind **user1** from **tenant1** and then associate **user1** with **tenant2**.
- If Kerberos authentication is enabled for the cluster (the cluster is in security mode), associate a human-machine user with a Doris tenant. If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), associate a machine-machine user with the Doris tenant.
- If a common Doris user is associated (that is, the role of the user does not have the Doris administrator permission), the following error is reported when the user connects to Doris on the MySQL client, creates a table, and queries the table data:
  ERROR 1227 (42000): errCode = 2, detailMessage = Access denied; you need (at least one of) the USAGE/ADMIN privilege(s) to use workload group '*Tenant name*'.

  Connect to Doris on the MySQL client as a user with the Doris administrator permission and run the following command to grant permission to the current user:

  GRANT USAGE_PRIV ON workload group '*Tenant name* **' to '***User ame* **';**

5.7.3.3 Multi-Tenancy Alarms

Doris multi-tenancy is built on top of the Workload Group Resource soft limit of the kernel. Workload groups only restrict the compute and memory resources used by tasks in a group on a single BE node. There is no resource pool for tenants. When query tasks are executed, resources are dynamically allocated to each BE node.

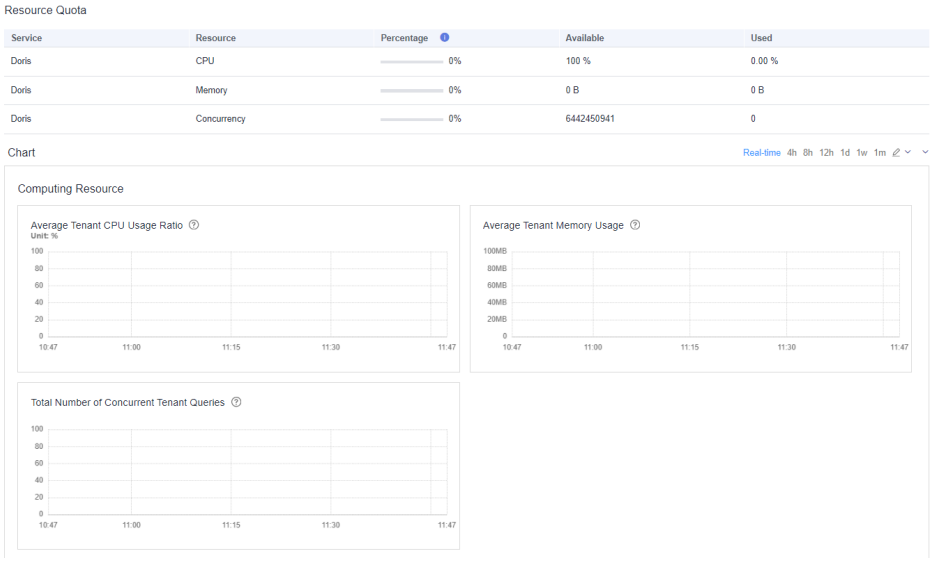
Doris multi-tenancy alarms are reported for nodes. The monitoring metric data can be aggregated by service or tenant.

Multi-Tenancy Monitoring

- Tenant monitoring

  On the FusionInsight Manager homepage, click **Tenant Resources**. In the tenant list, click the name of a Doris tenant. In the **Summary** tab, you can view monitoring information of the tenant in the **Resource Quota** and **Chart** areas.

  - CPU: average CPU usage of all BE nodes used by the tenant
  - Memory: average memory usage of all BE nodes used by the tenant
  - Concurrency: total number of concurrent queries on all FE nodes used by the tenant

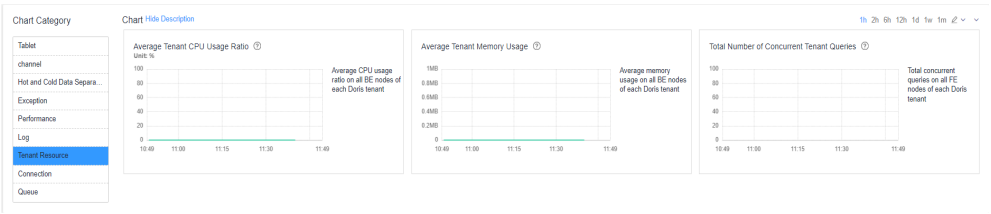


NOTE

- Resource quota is not refreshed in real time. The resource usage is queried only when you go to the **Summary** tab. The monitoring is in real time but monitoring data is refreshed every 30 seconds.
- The **Average Tenant CPU Usage Ratio** chart shows the average percentage of the time when the query tasks of the tenant occupy the CPU resources of all BE nodes.
- The number of FEs is not counted when jobs are queued. Therefore, the number of concurrent queries set by a tenant takes effect only on FE nodes. The**Total Number of Concurrent Tenant Queries** chart indicates the overall number of concurrent queries of a tenant.

- Service monitoring

  On the FusionInsight Manager homepage, choose **Cluster** > **Services** > **Doris** and click **Chart**. In **Chart Category**, select **Tenant Resource** to view the resource usage of all Doris tenants.



- Instance monitoring

  On the FusionInsight Manager homepage, choose **Cluster** > **Services** > **Doris** and click **Instances**, click the FE or BE instance of the desired tenant. In the **Chart** tab, select **Tenant Resource** from **Chart Category**. You can view the node resources used by all tenants.

Figure 5-3 FE instance resource monitoring

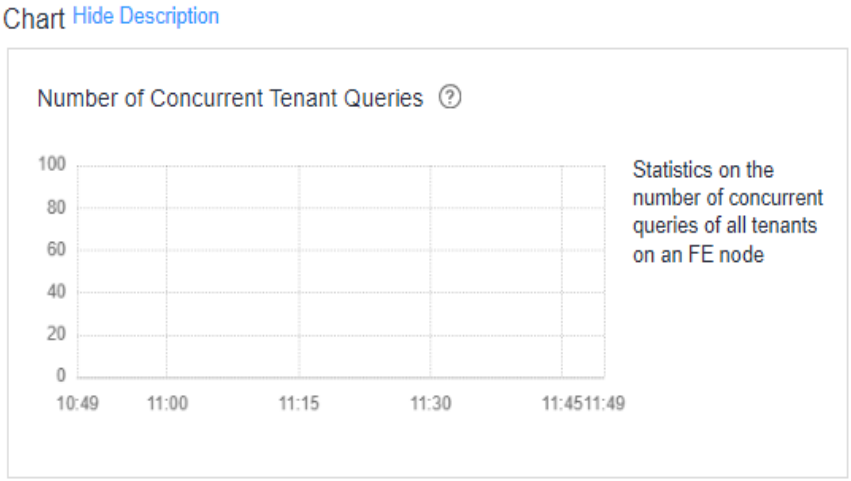
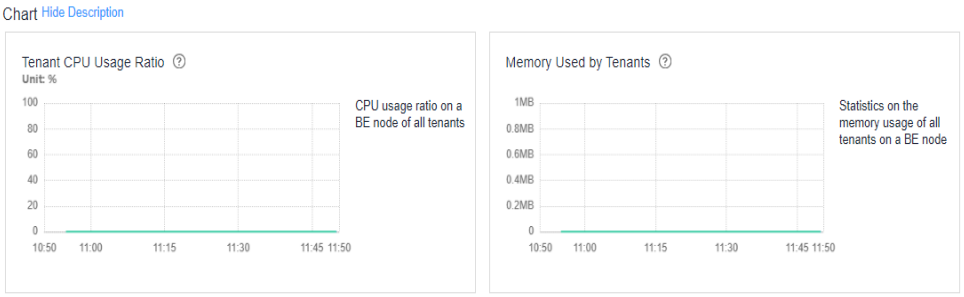


Figure 5-4 BE instance resource monitoring



Multi-Tenancy Alarms

Doris multi-tenancy alarms include:

- Number of concurrent requests (alarm ID: 50227): The alarm is generated when the number of concurrent tenant requests on an FE node exceeds the threshold (90% by default).
- Memory usage (alarm ID: 50228): The alarm is generated when the memory usage of a BE node exceeds the threshold (90% for critical alarms and 85% for major alarms).

NOTE

- The memory usage alarm is generated only for tenants who disabled soft memory soft limit.
- On FusionInsight Manager, choose **O&M** > **Alarm** > **Alarms**, click on the left of an alarm name, and determine the role and node for which the alarm is generated in the **Location** field. To identify the tenant for which the alarm is generated, view the monitoring chart on the FE or BE node for which the alarm is generated.

5.7.4 Doris Cold and Hot Data Separation

5.7.4.1 Introduction

For data analysis, it is common to query hot and cold data at varying frequencies and response speed. For example, to analyze user behavior, traffic data must be frequently queried and and the requests need to be quickly responded. Historical data that is seldomly accessed needs to be backed up over a long period for audit and backtracking. Query demands on such data decrease sharply as time goes by. If all data is stored locally, a large number of resources will be wasted.

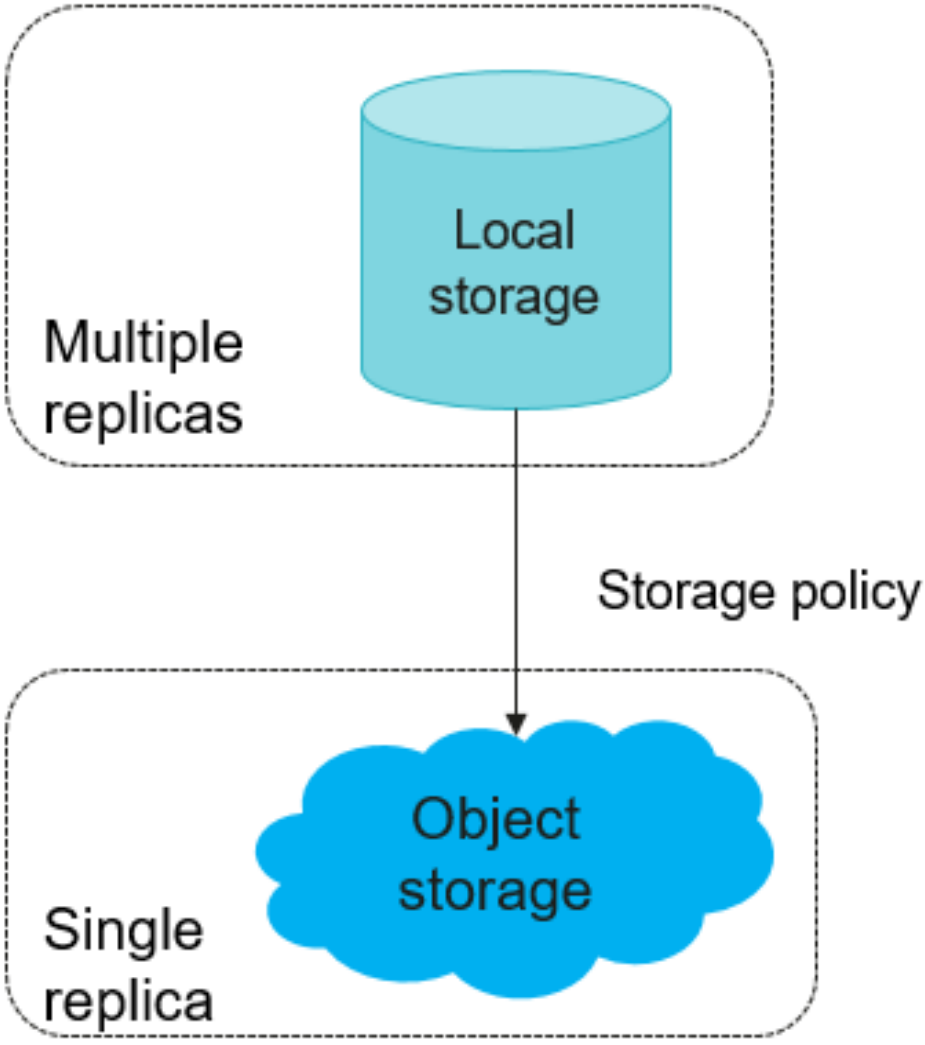
NOTE

This topic is available for MRS 3.3.1 or later only.

Principles

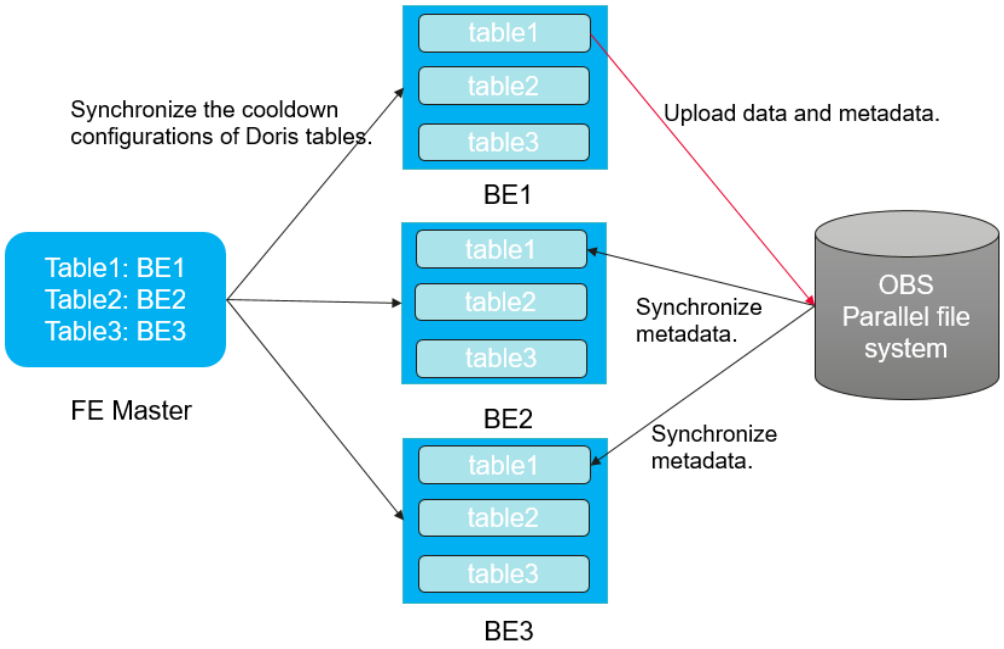
Apache Doris 2.0 supports the separation storage of cold and hot data. You can use this function to sink data from the local host to the object storage.

Figure 5-5 Hot and cold data separation



OBS provides secure, reliable, and cost-effective distributed storage service that supports large-scale data. Doris uses OBS to store data separately. **Figure 5-6** shows the principle.

Figure 5-6 Cold and hot data separation principle



5.7.4.2 Storing Cold and Hot Data Separately in Doris

This topic describes how to configure cold and hot data separation in Doris.

Prerequisite

The Doris cluster can communicate with OBS. For details, see **Initial Configuration**.

Creating an OBS Parallel File System and Obtaining the AK/SK and Domain ID

Create an OBS parallel file system.

Step 1 Log in to the OBS console.

Step 2 Choose **Parallel File Systems** > **Create Parallel File System**.

Step 3 Enter a file system name, for example, **doris-obs**.

The name of an enterprise project must be the same as that of the MRS cluster. Set other parameters.

Step 4 Click **Create Now**.

Step 5 In the parallel file system list, click the name of the one you just created and click **Overview** to obtain the endpoint information.

NOTE

After a service is deleted or a cluster is uninstalled, dirty data may remain in the parallel file system created in **Step 2** to **Step 4**. You need to delete the dirty data.

Obtain AK/SK information.

Step 6 Move the cursor on the username in the upper right corner, and select **My Credentials** from the drop-down list.

Step 7 On the **API Credentials** page, obtain the **Account ID** whish is used as the domain ID.

Step 8 Click **Access Keys**, click **Create Access Key**, and enter the verification code or password. Click **OK** and download the access keys. Obtain the AK/SK information from the **.csv** file.

----End

Creating a Cloud Service Agency and Binding It to a Cluster

Step 1 Log in to the MRS management console.

Step 2 In the service list, choose **Management & Governance** > **Identity and Access Management**.

Step 3 In the navigation pane on the left, choose **Agencies** and click **Create Agency**. On the displayed page, set the following parameters and click **Next**:

- **Agency Name**: Enter an agency name, for example, **mrs_ecs_obs**.
- **Agency Type**: Select **Cloud service**.
- **Cloud Service**: Select **Elastic Cloud Server (ECS) and Bare Metal Server (BMS)**.
- **Validity Period**: Select **Unlimited**.

Step 4 On the displayed page, search for the **OBS OperateAccess** policy and select it.

Step 5 Click **Next**, click **Show More**, select **Global services**, and click **OK**.

Step 6 In the displayed dialog box, click **OK** to start authorization. Click **Finish** after the message "Authorization successful." is displayed.

Step 7 On the MRS console, choose **Active Clusters** in the navigation pane on the left and click the name of the target cluster to access its details page.

Step 8 On the **Dashboard** page, click **Synchronize** on the right of **IAM User Sync** to synchronize the IAM user.

Step 9 Click **Manage Agency** on the right of **Agency**, select the created agency, for example, **mrs_ecs_obs**, and click **OK** to bind the agency to the cluster.

----End

Creating a Common Account Agency and Binding It to a Cluster

Step 1 Log in to the management console.

Step 2 In the service list, choose **Management & Governance** > **Identity and Access Management**.

Step 3 Choose **Permissions** > **Policies/Roles**, and click **Create Custom Policy**. Set the following parameters, and click **OK**:

- **Policy Name**: Enter a policy name, for example, **doris-policy**.
- **Policy View**: Select **JSON**.
- **Policy Content**: Enter the following content.

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Effect":"Allow",
         "Action":[
            "OBS:*:*"
            ]
      }
      ]
}
```

Step 4 In the navigation pane on the left, click **Agencies**. Click **Create Agency**. On the displayed page, set the following parameters and click **Next**:

- **Agency Name**: Enter an agency name, for example, **agency-MRS-to-OBS**.
- **Agency Type**: Select **Account**.
- Enter your cloud account in the **Delegated Account** field, that is, the account you register using your mobile phone number. It cannot be a federated user or an IAM user created using your cloud account.
- **Validity Period**: Select **Unlimited**.

Step 5 In the search box on the displayed **Authorize Agency** page, search for the custom policy created in **Step 3** and select it, for example, **doris-policy**.

Step 6 Click **Next**, click **Show More**, select **Global services**, and click **OK**.

Step 7 Check and record the agency ID.

Step 8 On the **Identity and Access Management** page, click **Agencies**.

Step 9 Click the name of the cloud service agency created in **Step 3**, for example, **mrs_ecs_obs**.

Step 10 Click the **Permissions** tab and click **Authorize**. On the displayed page, click **Create Policy** in the upper right corner, and set the parameters as follows:

- **Policy Name**: Enter a policy name, for example, **doris-assume-policy**.
- **Policy View**: Select **JSON**.
- **Policy Content**: Enter the following content. *{Agency ID}* indicates the ID recorded in **Step 7**.

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Action": [
            "iam:agencies:assume"
         ],
         "Resource": {
            "uri": [
               "/iam/agencies/{Agency ID}"
            ]
         },
         "Effect": "Allow"
      }
   ]
}
```

Step 11 Click **Next**. On the **Select Policy/Role** page, select the policy created in **Step 10**.

Step 12 Click **Next**, click **Show More**, select **Global services**, and click **OK**.

----End

Enabling Cold and Hot Data Separation

By default, cold and hot data separation is disabled. To use this function, perform the following operations:

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris**. Click the **Configurations** tab.

Step 2 Search for the **obs_cooldown_enable** parameter and set it to **true**.

Step 3 (Optional) If the data on the local disk is cooled down and stored on OBS, and related data needs to be stored on the local disk in a certain period of time, select **All Configurations** > **BE(Role)** > **Customization**, add the **moveback_enable** parameter to the customized parameter **be.conf.customized.configs** and set the parameter value to **true**.

Step 4 Click **Save** and then **OK**.

Step 5 Click **Instances**, select the affected FE and BE instances, choose **More** > **Restart Instance**, and enter the password of the current user to restart the FE and BE instances.

----End

Use Case

Step 1 Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login user* **-p -P***Connection port for FE queries* **-h***IP address of the Doris FE instance*

Enter the password for logging in to the database.

NOTE

- To obtain the query connection port of the Doris FE instance, you can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the service IP address of any FE instance.
- You can also use the MySQL connection software or Doris web UI to connect to the database.

Step 2 Create a resource.

- Create a resource that supports an agency for OBS authentication.
  a. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris**. Click **Configurations** then **All Configurations**, choose **FE(Role)** > **OBS**, and modify the following parameters and save the changes:
     - **obs_authentication_method**: Change the value to **agency**.
     - **obs_endpoint**: endpoint information queried in **Step 5**, for example, **obs.***XXX*.
     - **object_storage_security_provider**: Retain the default value **com.huawei.mrs.MrsObsCredentialsProvider**.
  b. On the **Dashboard** page, click **More** > **Restart Service**. Enter the password of the current user and click **OK** to restart the Doris service.
  c. Connect the node where the MySQL client is installed to Doris. For details, see **Using the MySQL Client to Connect to Doris**.
  d. Run the following statement to create a resource:

     CREATE RESOURCE IF NOT EXISTS *resource_obs_hot_cold* **PROPERTIES (**

     "type" = "obs",

     "obs.bucket" = "*Name of the OBS parallel file system created in **Step 3***",**

     "obs.root.path" = "*Root directory for storing data***"**

     "obs.region" = " *Region of the cluster where the Doris service is deployed***"**

     );
- Create a resource that supports permanent OBS authentication.
  a. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris**. Click **Configurations** then **All Configurations**, choose **FE(Role)** > **OBS**, and modify the following parameters and save the changes:
     - **obs_authentication_method**: Change the value to **permanent**.

- **object_storage_security_provider**: Retain the default value **com.huawei.mrs.MrsObsCredentialsProvider**.

b. On the **Dashboard** page, click **More** > **Restart Service**. Enter the password of the current user and click **OK** to restart the Doris service.

c. Connect the node where the MySQL client is installed to Doris. For details, see **Using the MySQL Client to Connect to Doris**.

d. Run the following statement to create a resource:

CREATE RESOURCE IF NOT EXISTS *resource_obs_hot_cold* **PROPERTIES (**

"type" = "obs",

"obs.endpoint" = "*xxx***",**

"obs.region" = "*xxx***",**

"obs.bucket" = "*xxx***",**

"obs.root.path" = "*xxx***",**

"obs.access_key" = "*xxx***",**

"obs.secret_key" = "*xxx***"**

);

NOTE

- **type**: data storage type. The value is **obs**.
- **obs.endpoint**: endpoint information viewed in **Step 5**
- **obs.region**: region of the cluster where the Doris service is deployed
- **obs.bucket**: name of the OBS parallel file system created in **Step 3**
- **obs.root.path**: root directory for storing data
- **obs.access_key**: AK information obtained in **Step 8**
- **obs.secret_key**: SK information obtained in **Step 8**

Step 3 Set the data cooling policy using the storage policy.

- Set the time to live (TTL) to cool down data.

  CREATE STORAGE POLICY IF NOT EXISTS *policy_doris_hot_cold* **PROPERTIES("storage_resource" = "***resource_obs_hot_cold***", "cooldown_ttl" = "***1d***");**

  If the value of **cooldown_ttl** is **1d**, newly imported data will be cooled one day later and the cooled data will be stored in the OBS path configured during resource creation in **Step 2**.

- Set a time point to cool down data.

  In addition to setting the TTL, you can also set a time point in the cooling policy.

  CREATE STORAGE POLICY IF NOT EXISTS *policy_doris_hot_cold2* **PROPERTIES("storage_resource" = "resource_obs_hot_cold", "cooldown_datetime" = "***2024-01-01 10:00:00***");**

Step 4 Set the storage policy of a table or partition.

- Set the storage policy when creating a table.

  CREATE TABLE *ORDERS* **(**

  ORDER_ID VARCHAR(50),


```
USER_ID BIGINT,
PRODUCT_ID VARCHAR(10),
PRICE DECIMAL(15,2),
CHANNEL VARCHAR(20),
CREATE_DT DATE
)
DUPLICATE KEY(`ORDER_ID`)
PARTITION BY RANGE(`CREATE_DT`)
(
PARTITION `p202401` VALUES LESS THAN ("2024-02-01"),
PARTITION `p202402` VALUES LESS THAN ("2024-03-01")
)
DISTRIBUTED BY HASH(`ORDER_ID`) BUCKETS 3
PROPERTIES (
"replication_num" = "3",
"storage_policy" = "policy_doris_hot_cold "
);
```

- Modify the properties of an existing table.

  ALTER TABLE *ORDERS* **SET("storage_policy" = "policy_doris_hot_cold");**

- Set the cold and hot separation policy for a partition when creating a table.

```
CREATE TABLE ORDERS (
ORDER_ID VARCHAR(50),
USER_ID BIGINT,
PRODUCT_ID VARCHAR(10),
PRICE DECIMAL(15,2),
CHANNEL VARCHAR(20),
CREATE_DT DATE
)
DUPLICATE KEY(`ORDER_ID`)
PARTITION BY RANGE(`CREATE_DT`)
(
PARTITION `p202401` VALUES LESS THAN ("2024-02-01")
("storage_policy" = "policy_doris_hot_cold"),
PARTITION `p202402` VALUES LESS THAN ("2024-03-01")
)
DISTRIBUTED BY HASH(`ORDER_ID`) BUCKETS 3
PROPERTIES (
"replication_num" = "3"
);
```

- Modify the partition properties of an existing table.

ALTER TABLE *ORDERS* **MODIFY PARTITION (`p202401`) SET("storage_policy"="policy_doris_hot_cold");**

NOTE

- A single table or partition can be associated with only one storage policy. Associated storage policies cannot be deleted before disassociation.
- Information about the object associated with a storage policy cannot be modified, such as bucket, endpoint, and root_path.
- A storage policy can be created, modified, and deleted. Before deleting a storage policy, ensure that no table references the storage policy.
- When the Merge-on-Write feature is enabled for the Unique model, storage policies cannot be set.

Step 5 Run the following statement to query data:

show tablets from *ORDERS*;

This command views the tablet information of the table. In the tablet information, LocalDataSize and RemoteDataSize are distinguished. LocalDataSize indicates the data stored locally, and RemoteDataSize indicates the data that has been cooled and stored on OBS.

Before the data is cooled, the tablet information of the table is as follows.

```
*************************** 1. row ***************************
              TabletId: 10210
             ReplicaId: 10211
             BackendId: 10002
            SchemaHash: 1629210097
               Version: 26
      LstSuccessVersion: 26
       LstFailedVersion: -1
          LstFailedTime: NULL
         LocalDataSize: 2718
        RemoteDataSize: 0
              RowCount: 25
                 State: NORMAL
LstConsistencyCheckTime: NULL
          CheckVersion: -1
          VersionCount: 2
             QueryHits: 0
              PathHash: -2692135266043659989
               MetaUrl: http://192.165.0.218:29986/api/meta/header/10210
      CompactionStatus: http://192.165.0.218:29986/api/compaction/show?tablet_id=10210
      CooldownReplicaId: 10211
        CooldownMetaId: TUniqueId(hi:-629967361863696654, lo:-8577590943874589763)
1 row in set (0.00 sec)
```

After the data is cooled, the tablet information of the table is as follows.


```
*************************** 1. row ***************************
              TabletId: 10210
             ReplicaId: 10211
             BackendId: 10002
            SchemaHash: 1629210097
               Version: 26
        LstSuccessVersion: 26
         LstFailedVersion: -1
            LstFailedTime: NULL
           LocalDataSize: 0
          RemoteDataSize: 2718
                RowCount: 25
                   State: NORMAL
LstConsistencyCheckTime: NULL
            CheckVersion: -1
            VersionCount: 2
               QueryHits: 0
                PathHash: -2692135266043659989
                 MetaUrl: http://192.165.0.218:29986/api/meta/header/10210
        CompactionStatus: http://192.165.0.218:29986/api/compaction/show?tablet_id=10210
        CooldownReplicaId: 10211
           CooldownMetaId: TUniqueId(hi:-629967361863696654, lo:-8577590943874589763)
1 row in set (0.01 sec)
```

----End

5.7.5 Doris Slow Query Detection

Scenario

As a ROLAP database, Doris relies on a powerful relational database engine at its foundation. High concurrency and aggregation queries are commonly used. MRS Doris lacks slow query detection capabilities, making it difficult to monitor query execution for troubleshooting. To improve Doris's O&M capabilities, we developed the slow query detection function.

Doris slow query information can be displayed and managed on Manager only after this function enabled. By default, this function is disabled to reduce the impact on environment resources. You can enable this function with custom parameter **query_history_enable**.

This function is available only for MRS 3.5.0 and later versions.

Prerequisites

- A cluster containing the Doris service has been created, and all services in the cluster are running properly.
- The nodes to be connected to the Doris database can communicate with the MRS cluster.
- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.

Enabling Slow Query Detection

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris** > **Configurations** > **All Configurations** > **FE(Role)** > **Customization**.

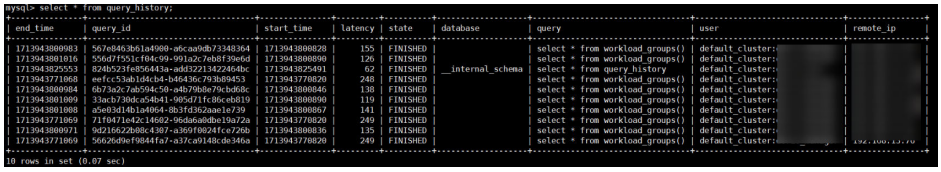
Step 2 Add the **query_history_enable** parameter to the custom parameter **fe.conf.customized.configs** and set the parameter value to **true**.

Step 3 Click **Save**, and then click **OK** to save the configurations.

Step 4 Click **Instances**, select the affected FE instances, choose **More** > **Restart Instance**. Enter the password of the user and click **OK** to apply the configuration.

Step 5 After connecting the MySQL client to Doris, run the following command to view the slow queries executed by the Doris:

select * from __internal_schema.query_history;



Step 6 On FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Queries** to view **Ongoing Slow Queries** and **Completed Slow Queries**. You can also view the top 10 users who execute slow queries, IP addresses of the clients that submit slow SQL statements, and slow query statements.

- **Top 10 Users**: top 10 Doris users who perform slow queries
- **Top 10 IP Addresses**: IP addresses of top 10 Doris clients that execute slow queries
- **Top 10 SQL Statements**: top 10 most executed slow query SQL statements on the Doris client

----End

Ongoing Slow Queries

You can query information about slow SQL statements that are being executed but do not return any result.

- **Access path**

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Queries**, and click the **Ongoing Slow Queries** tab. For details about the parameters, see **Table 5-9**.

- **Filter conditions**

  Select the query condition as required and filter the query results.

  Table 5-8 Filter conditions

Conditions	Parameter Description
Slow query duration exceeding	Filters the slow SQL queries based on the duration. The value can be **3 (s)**, **9 (s)**, **15 (s)**, or **25 (s)**.
By Query ID	Filters the slow queries based on the query ID. Fuzzy search based on the query ID is supported. For example, if the query ID is **111-222-333-444-555**, you can enter part of it such as **111-222** or **-222-333** to query the needed information.

Conditions	Parameter Description
By User	Filters the slow queries based on Doris users who execute the SQL statements. Fuzzy search based on part of a username is supported.
By Client IP Address	Filters the slow queries based on the IP address of the client that submits slow query SQL statements. Fuzzy search based on part of the client IP address is supported. For example, if the client IP address is **192.168.0.1**, you can enter part of it such as **192.168** or **192.168.0** to query the information you need.

Completed Slow Queries

You can query information about slow query SQL statements that have been executed and returned results.

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris**. On the displayed page, click the **Query Management** tab and then the **Completed Queries** tab.

For details about slow query parameters and filter conditions, see **Table 5-9** and **Table 5-8**, respectively.

Slow Query Detection Parameters

Table 5-9 Slow query parameters

Parameter	Description
Query ID	Unique ID generated internally.
Query	Slow query SQL statement.
Start Time	Time when the query starts.
End Time	Time when the query is complete.
Database	Name of the database where the table in the query is.
Instance Name	FE instance node that executes the query. It is displayed for **Ongoing Slow Queries** only.
Memory Used	Memory space consumed by the query, in bytes. This field is displayed for **Completed Queries** only.
Duration	Total execution time of the query, in seconds.
User	Doris user who executes the query.
Client IP Address	IP address of the client that submits the query.

Parameter	Description
Status	Execution status of the query. Possible values are as follows: • **Running**: queries being executed • **Cancelled**: slow queries stopped by clicking **Terminate** in the **Operation** column in the **Ongoing Slow Queries** tab • **Completed**: slow queries that have been properly executed
Operation	Column available for **Ongoing Slow Queries** only. You can click **Terminate** to stop a query.

5.8 Doris O&M Management

5.8.1 Doris Log Overview

Log Description

Log path: The default storage path of Doris logs is **/var/log/Bigdata/doris/***role name*.

- FE: **/var/log/Bigdata/doris/fe** (run logs) and **/var/log/Bigdata/audit/doris/fe** (audit logs)
- BE: **/var/log/Bigdata/doris/be** (run logs)
- DBroker: **/var/log/Bigdata/doris/dbroker** (run logs)

Log archive rule: The automatic compression and archive function is enabled for Doris logs. By default, when the size of a log file exceeds the specified size (configurable), the log file is automatically compressed. The naming rule of the compressed log file is as follows: *Original log file name* **>-**<*yyyy-mm-dd_hh-mm-ss*>**.** *[ID]***.log.zip**. A maximum of 20 latest compressed files are retained. The number of compressed files and compression threshold can be configured.

Table 5-10 Hive log list

Log type	Log File	Description
Run log	/fe/fe.out	Standard/Error output logs (stdout and stderr)
	/fe/fe.log	Main log, including all contents except fe.out
	/fe/fe.warn.log	Subset of fe.log. Only WARN and ERROR logs are recorded.

Log type	Log File	Description
	/fe/fe-omm-<Date>-<PID>-gc.log.<No.>	GC logs of the FE process
	/fe/preStart.log	Work logs before the FE is started
	/fe/check_fe_status.log.log	Log file that records whether the FE service is started successfully
	/fe/cleanup.log	Cleanup log file for FE uninstallation
	/fe/start_fe.log	FE process startup logs
	/fe/stop_fe.log	FE process stop log
	/fe/postinstallDetail.log	Work logs generated after the FE is installed and before the FE is started
	/be/be.INFO	Run log of the BE process
	be.WARNING	Subset of be.log. Only WARN and FATAL logs are recorded.
	/be/be-omm-<Date>-<PID>-gc.log.<No.>	GC logs of the BE process
	/be/postinstallDetail.log	Work logs generated after BE installation and before BE startup
	/be/preStart.log	Work logs before BE startup
	/be/cleanup.log	Cleanup log file for BE uninstallation
	/be/start_be.log	BE process startup logs
	/be/stop_be.log	BE process stop log
	/be/check_be_status.log	Log file that records whether the BE service is started successfully
	/be/be.out	Standard/Error output logs of the BE process (stdout and stderr)
	/dbroker/start_broker.log	Logs indicating that the DBroker process is started or stopped properly
	/dbroker/stop_broker.log	DBroker process startup and stop exception logs
	/dbroker/preStart.log	Work log before the DBroker is started
	/dbroker/cleanup.log	Cleanup log file generated during or before DBroker uninstallation

Log type	Log File	Description
	/dbroker/check_db_status.log	Log file that records whether the DBroker service is started successfully
	/dbroker/dbroker-omm-<Date>-<PID>-gc.log.<No.>	GC log of the DBroker process
	/dbroker/apache_hdfs_broker.log	Run log of the DBroker process
Viewing Audit Logs	fe.audit.log	Audit log, which records all SQL requests received by the FE

Log Levels

Table 5-11 describes the log levels provided by the Doris.

The priorities of run log levels are FATAL, ERROR, WARN, and INFO in descending order. Logs whose levels are higher than or equal to the set level are printed. The number of printed logs decreases as the set log level increases.

Table 5-11 Log Levels

Level	Description
FATAL	FATAL usually indicates a program assertion error.
ERROR	Logs of this level record error information about system running
WARN	Logs of this level record exception information about the current event processing
INFO	Logs of this level record normal running status information about the system and events

To change log levels, perform the following operations:

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris** > **Configurations** > **All Configurations**. The **All Configurations** page of the Doris service is displayed.

Step 2 On the menu bar on the left, select the log menu of the target role.

Step 3 Select a desired log level and save the configuration.

Step 4 Click **DashBoard**, choose **More** > **Restart Service**, and enter the password of the current user to restart the Doris service.

----End

Log Formats

The Doris log format is as follows:

Table 5-12 Log format

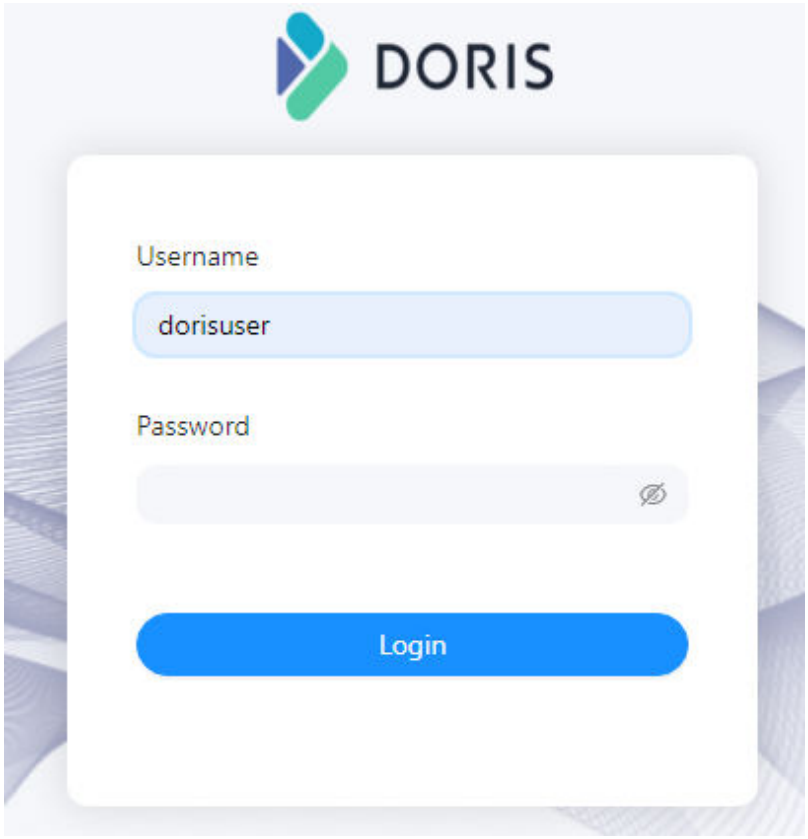
Log type	Format	Example Value
FE run log	<yyyy-MM-dd HH:mm:ss,SSS><LogLevel>(Thread name|Thread ID)<Location where the log event occurs>|<message in log>	2023-04-13 11:17:14,371 INFO (tablet stat mgr|34) [TabletStatMgr.runAfterCatalogReady():125] finished to update index row num of all databases. cost: 0 ms
BE run logs	Log level. I indicates INFO, W indicates WARN, and F indicates FATAL|MMdd HH:mm:ss.SSS. Thread ID. Location where a log event occurs. message&gt in log;	I0413 11:26:03.439189 25248 tablet_manager.cpp:895] begin to build all report tablets info
DBroker run log	<MMdd HH:mm:ss.SSS> <Thread ID> <LogLevel><message in log>	2023-04-11 11:43:13 [main:0] - [INFO] starting apache hdfs broker....

Log type	Format	Example Value
Viewing Audit Logs	<yyyy-MM-dd HH:mm:ss,SSS[Operation type] >|<Client>|<User Name>|<Db Name>|<State>|<ErrorCode>|<ErrorMessage>|<Time>|<ScanBytes>|<ScanRows>|<ReturnRows>|<StmtId>|<QueryId>|<IsQuery>|<feIp>| |<Stmt>|<CpuTimeMS>|<SqlHash>|<peakMemoryBytes>|<SqlDigest>|<TraceId>|<FuzzyVariables>	2023-04-13 10:49:26,410 [query] | Client=192.168.64.223:44382| User=root|Db=hivedoris| State=ERR|ErrorCode=1105| ErrorMessage=errCode = 2, detailMessage = (192.168.64.78) [INTERNAL_ERROR]failed to init reader for file /user/hive/warehouse/hivedoris.db/test/000000_0, err: [INTERNAL_ERROR]connect to hdfs failed. error: (255), Unknown error 255), reason: NullPointerException: | Time=67|ScanBytes=0| ScanRows=0|ReturnRows=0| StmtId=91| QueryId=e1125283f12c4994-a69e3a323044d681| IsQuery=true| feIp=192.168.64.78| Stmt=select * from test| CpuTimeMS=0| SqlHash=3bbc220823c3e757002fb9490196cf84| peakMemoryBytes=0| SqlDigest=|TraceId=| FuzzyVariables=

5.8.2 Accessing the Doris Web UI for Component Status

Step 1 Log in to FusionInsight Manager as a user with administrator rights. Choose **Cluster** > **Services** > **Doris**.

Step 2 On the **Dashboard** page, click the hyperlink on the right of **FE WebUI**. On the displayed Doris web UI login page, enter the username and password of the account with the Doris management permission. The initial password must have been changed if Kerberos authentication is enabled for the cluster (the cluster is in security mode). For details about how to create a user, see **Creating a Doris Permission Role**. Then, click **Login**.



Step 3 On the home page of the Doris web UI, view the Doris cluster information. You can also view the Doris table information in **Playground** and run a SQL query statement.

----End

5.8.3 Backing Up Doris Data

Doris can back up current data as files to a remote storage system through Broker. Doris data can be periodically backed up using snapshots and migrated.

NOTE

- Currently, only users with the **ADMIN** permission can perform backup and restoration operations.
- Only one backup job can be being executed in a database.
- Doris data backup supports operations at the minimum partition level. When a table contains a large amount of data, you are advised to perform operations by partition to reduce the cost of retry upon failure.
- The backup and restoration operations are performed on the actual data files. Therefore, when a table has too many shards or a shard has too many minor versions, the backup may take a long time even if the total data volume is small.
- When you run the **SHOW BACKUP** or **SHOW RESTORE** command to view the job status, error information may be displayed in the TaskErrMsg column. If the value in the **State** column is not **CANCELLED**, the job continues. These tasks may be retried successfully. However, some tasks are incorrect, causing job failures.

Data Backup Principles

The backup operation is to upload the data of a specified table or partition to the remote repository as a file stored in the Doris. After a user submits a backup request, the system performs the following operations:

1. Snapshot and Snapshot Upload

   Backup is performed on snapshots. In the snapshot phase, snapshots are taken for specified tables or partition data files. A snapshot only generates a hard link for the current data file, which is time-consuming. After a snapshot is created, operations such as modification and import on the table do not affect the backup result. After the snapshot is complete, the system starts to upload the snapshot files one by one. The snapshot files are concurrently uploaded by each backend.

2. Preparing and Uploading Metadata

   After the data file snapshot is uploaded, Frontend writes the corresponding metadata into a local file and uploads the local metadata file to the remote repository through Broker to complete the backup job.

NOTE

- If the table to be backed up is a dynamic partition table, the dynamic partition attribute is automatically disabled after the backup. Before restoring data, run the following command to manually enable the dynamic partition attribute of the table:

  ALTER TABLE *tbl1* **SET ("dynamic_partition.enable"="true")**

- The data backup operation does not retain the **colocate_with** attribute of the table.

Prerequisite

- A cluster containing the Doris service has been created, and all services in the cluster are running properly.
- The node to be connected to the Doris database can communicate with the MRS cluster.
- The MySQL client has been installed. For details about related operations, see **Using the MySQL Client to Connect to Doris**
- Create a user with the Doris management permission.
  - Kerberos authentication is enabled for the cluster (the cluster is in security mode)

    On FusionInsight Manager, create a human-machine user, for example, dorisuser, create a role with the Doris administrator rights, and bind the role to the user.

    Log in to FusionInsight Manager as the new user **dorisuser** and change the initial password of the user.
  - Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

    After connecting to Doris as user **admin**, create a role with administrator permissions, and bind the role to the user.

Backing Up Doris Data

Step 1 Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login username* **-p***Database login password* **-P***Database connection port* **-h***IP address of the Doris FE instance*

NOTE

- The database connection port is the query connection port of the Doris FE. You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris WebUI to connect to the database.

Step 2 Run the following command to create a remote repository **example_repo** in HDFS:

- Kerberos authentication is enabled for the cluster (the cluster is in security mode)

  CREATE REPOSITORY `example_repo`

  WITH BROKER `hdfs_broker`

  ON LOCATION "hdfs://hadoop-name-node:25000/path/to/repo/"

  PROPERTIES

  (

  "hadoop.security.authentication"="kerberos",

  "kerberos_principal"="doris/hadoop.hadoop.com@HADOOP.COM",

  "kerberos_keytab"="/opt/huawei/Bigdata/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-fe/bin/doris.keytab"

  );

- Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

  CREATE REPOSITORY `example_repo`

  WITH BROKER `hdfs_broker`

  ON LOCATION "hdfs://hadoop-name-node:25000/path/to/repo/"

  PROPERTIES

  (

  "username" = "hdfs",

  "password" = ""

  );

Step 3 View the created repository.

SHOW REPOSITORIES;

Step 4 Back up data to the **example_repo**. You can back up table data or partition data. For example:

- To fully back up data in the **example_tbl** table in **example_db** to **example_repo**, run the following command:

  BACKUP SNAPSHOT example_db.snapshot_label1

  TO example_repo

  ON (example_tbl)

  PROPERTIES ("type" = "full");

- Fully back up the **p1** and **p2** partitions of the **example_tbl** table in the **example_db** and the **example_tbl2** table to the **example_repo**.

  BACKUP SNAPSHOT example_db.snapshot_label2

  TO example_repo

  ON

  (

  example_tbl PARTITION (p1,p2),

  example_tbl2

  );

Step 5 Run the following command to check the execution status of the backup job:

show BACKUP;

Step 6 Check whether the backup is successful in the remote repository.

SHOW SNAPSHOT ON example_repo WHERE SNAPSHOT = "snapshot_label1";

```
+-----------------+---------------------+--------+
| Snapshot        | Timestamp           | Status |
+-----------------+---------------------+--------+
| snapshot_label1 | 2022-04-08-15-52-29 | OK     |
+-----------------+---------------------+--------+
1 row in set (0.15 sec)
```

----End

5.8.4 Restoring Doris Data

Doris can back up current data as files to a remote storage system through Broker. Run the restoration command to restore data from the remote storage system to any Doris cluster. Doris data can be periodically backed up using snapshots and migrated.

NOTE

- Currently, only users with the **ADMIN** permission can perform backup and restoration operations.
- Only one recovery job can be being executed in a database.
- Doris data restoration supports operations at the minimum partition level. When a table contains a large amount of data, you are advised to perform operations by partition to reduce the cost of retry upon failure.
- The backup and restoration operations are performed on the actual data files. Therefore, when a table has too many shards or a shard has too many minor versions, it may take a long time to restore the table even if the total data volume is small.
- When you run the **SHOW BACKUP** or **SHOW RESTORE** command to view the job status, error information may be displayed in the TaskErrMsg column. If the value in the **State** column is not **CANCELLED**, the job continues. These tasks may be retried successfully. However, some tasks are incorrect, causing job failures.
- If the recovery job is an overwrite operation (specifying that data is restored to an existing table or partition), the overwritten data in the current cluster may not be restored from the COMMIT phase of the recovery job. If the recovery job fails or is canceled, the previous data may be damaged and cannot be accessed. In this case, you need to perform the restoration operation again and wait until the job is complete. Therefore, you are not advised to restore data by overwriting data unless you confirm that the current data is no longer used.

Data Restoration Principles

To restore Doris data, you need to specify an existing backup data in a remote repository and then restore the backup data to the local cluster. After a restore request is submitted, the system performs the following operations:

1. Create the corresponding metadata locally.

   Structures such as table partitions are created and restored in the local cluster. After the table is created, it is visible but inaccessible.

2. Local snapshot

   Create a snapshot for the table created in the local cluster. The snapshot is an empty snapshot (the newly created table has no data). The snapshot is used to generate the corresponding snapshot directory on the Backend and receive the snapshot file downloaded from the remote repository.

3. Downloading a snapshot

   Snapshot files in the remote repository are downloaded to the corresponding snapshot directory and concurrently completed by each backend.

4. Making a Snapshot Take Effect

   After the snapshot is downloaded, map each snapshot to the metadata of the current local table. Then reload these snapshots for them to take effect and complete the final recovery job.

Prerequisite

- A cluster containing the Doris service has been created, and all services in the cluster are running properly.
- The node to be connected to the Doris database can communicate with the MRS cluster.
- The MySQL client has been installed. For details about related operations, see **Using the MySQL Client to Connect to Doris**

- Create a user with the Doris management permission.
  - Kerberos authentication is enabled for the cluster (the cluster is in security mode)

    On FusionInsight Manager, create a human-machine user, for example, dorisuser, create a role with the Doris administrator rights, and bind the role to the user.

    Log in to FusionInsight Manager as the new user **dorisuser** and change the initial password of the user.
  - Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

    After connecting to Doris as user **admin**, create a role with administrator permissions, and bind the role to the user.
- You have backed up the Doris table or partition data to be restored by referring to **Backing Up Doris Data**.

Restoring Doris Data

Step 1 Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login username* **-p***Database login password* **-P***Database connection port* **-h***IP address of the Doris FE instance*

NOTE

- The database connection port is the query connection port of the Doris FE. You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the IP address of the Doris FE instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the IP address of any FE instance.
- You can also use the MySQL connection software or Doris WebUI to connect to the database.

Step 2 Restore table or partition data from the remote warehouse where data has been backed up. For example:

- Run the following command to restore the **example_tbl** table in **snapshot_label1** from **example_repo** to the **example_db2** database. The table is restored to one backup.

  RESTORE SNAPSHOT *example_db2*.`*snapshot_label1*`

  FROM `*example_repo*`

  ON (`*example_tbl*` **)**

  PROPERTIES

  (

  "backup_timestamp"="*2023-08-16-20-13-55***",**

  "replication_num" = "*1***"**

);

You can run the **SHOW SNAPSHOT ON example_repo WHERE SNAPSHOT = "snapshot_label1";** command to obtain the value of backup_timestamp.

- Restore the partitions **p1** and **p2** of the **example_tbl** table in the backup **snapshot_label2** from **example_repo**, restore the **example_tbl2** table to the **example_db1** database, rename the table **new_tbl**. By default, three copies are restored.

  RESTORE SNAPSHOT *example_db1*.`*snapshot_2*`

  FROM `*example_repo*`

  ON

  (

  `*backup_tbl*` **PARTITION (**`*p1*`**,** `*p2*`**),**

  `*backup_tbl2*` **AS** `*new_tbl*`

)

  PROPERTIES

  (

  "backup_timestamp"="*2023-08-16-20-13-55***"**

  );

  Note: You can run the **SHOW SNAPSHOT ON example_repo WHERE SNAPSHOT = "snapshot_label2";** command to obtain the value of **backup_timestamp**.

Step 3 Run the following command to check the execution status of the recovery job:

SHOW RESTORE\G;

----End

5.8.5 Table of Audit Logs

This function enables direct SQL statement execution for viewing and analyzing audit logs, eliminating the need to manually collect and analyze FE audit log files to check service volume and types. You can periodically import FE audit logs into a specified Doris table using Stream Load. The audit log table function is disabled by default. You can use the **enable_audit_log_table** parameter to enable this function.

This function is available only for MRS 3.5.0 and later versions.

NOTE

- Currently, the audit log table does not record operations such as Broker Load, Export, and Stream Load. You need to view the operations in the audit log.
- To record the Stream Load operation with the audit log table, set the BE parameter **enable_stream_load_record** to **true**.
- The audit log table records only information about SQL statements that have been executed.
- By default, the maximum interval for writing data to the audit log table is 60 seconds, and the maximum amount of data that can be written in each batch is 50 MB. You can adjust the interval and amount by setting the **max_batch_interval_sec** and **max_batch_size** parameters.

Enabling Audit Log Table

Step 1 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations** > **All Configurations** > **FE(Role)** > **Log**, and change the value of **enable_audit_log_table** to **true**.

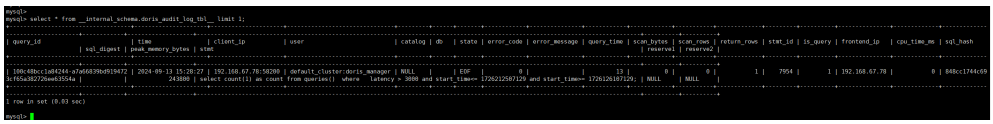
Step 2 Click **Save**, and then click **OK** to save the configurations.

Step 3 Click **Instances**, select the affected FE instances, choose **More** > **Restart Instance**. Enter the password of the user and click **OK** to apply the configurations.

Step 4 After the MySQL client is connected to Doris (for details, see **Getting Started with Doris**), run the following command to view the SQL statements that have been executed by Doris:

select * from __internal_schema.doris_audit_log_tbl__ limit 1;

Figure 5-7 SQL statements that have been executed



For details about each field, see **Table 5-13**.

----End

Fields in the Audit Log Table

Table 5-13 Audit log table fields

Field	Type	Description
query_id	varchar(48)	Query task ID.
time	datetime	Query start time.
client_ip	varchar(200)	IP address and port of the client.
user	varchar(64)	Name of the user who performs the query.
catalog	varchar(128)	Name of the catalog to which the query belongs.
db	varchar(96)	Name of the database to which the query belongs.
state	varchar(8)	Status of the query result.
error_code	int	Error code if the query fails

Field	Type	Description
error_message	string	Error message if the query fails
query_time	bigint	Query execution time, in ms.
scan_bytes	bigint	Total number of bytes scanned by the query.
scan_rows	bigint	Total number of rows scanned by the query.
return_rows	bigint	Number of rows returned in the query result.
stmt_id	int	Serially numbered statement ID.
is_query	tinyint	Whether the statement is a query statement. • **1**: indicates a query statement. • **0**: indicates a non-query statement.
frontend_ip	varchar(200)	IP address of the FE that executes the query.
cpu_time_ms	bigint	Total CPU time of the query.
sql_hash	varchar(48)	Hash value of the query.
sql_digest	varchar(48)	SQL summary. The value is empty for non-slow queries.
peak_memory_bytes	bigint	Peak memory usage of all BE nodes, in bytes.
stmt	string	Information about the SQL statement that is executed.
reserve1	string	Reserved field 1
reserve2	string	Reserved field 2

5.9 Typical SQL Syntax of Doris

5.9.1 CREATE DATABASE

This section describes the basic SQL syntax and usage of the Doris database.

Basic Syntax

CREATE DATABASE [IF NOT EXISTS] *db_name*

[PROPERTIES ("key"="value", ...)];

Use Example

Step 1 Connect to Doris through MySQL client as a user with Doris administrator permissions.

Step 2 Run the following command to create the database **example_db**:

create database if not exists example_db;

Step 3 Run the following command to view the database information:

SHOW DATABASES;

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| example_db         |
| information_schema |
+--------------------+
2 rows in set (0.00 sec)
```

Step 4 Run the following command to switch to **example_db**:

use example_db;

----End

5.9.2 CREATE TABLE

This section describes the basic SQL syntax and usage of Doris for creating tables.

Basic Syntax

CREATE TABLE [IF NOT EXISTS] [database.]*table*

(

column_definition_list,

[index_definition_list]

)

[engine_type]

[keys_type]

[table_comment]

[partition_info]

distribution_desc

[rollup_list]

[properties]

[extra_properties]

Use Example

- To create an ordinary table named **table1**, run the following command:

  CREATE TABLE example_db.table1
  (
  k1 TINYINT,
  k2 DECIMAL(10, 2) DEFAULT "10.5",
  k3 CHAR(10) COMMENT "string column",
  k4 INT NOT NULL DEFAULT "1" COMMENT "int column"
)
  COMMENT "table comment"
  DISTRIBUTED BY HASH(k1) BUCKETS 32;

- Create a partitioned table named **table2**.

  Use the **event_day** column as the partition column and create three partitions: p201706, p201707, and p201708. The values are as follows:

  - p201706: The value range is [Minimum value, 2017-07-01).
  - p201707: The value range is [2017-07-01, 2017-08-01).
  - p201708: The value range is [2017-08-01, 2017-09-01).

  Each partition uses **siteid** for hash bucketing. The number of buckets is 10.

  The command for creating a table is as follows:

  CREATE TABLE table2
  (
  event_day DATE,
  siteid INT DEFAULT '10',
  citycode SMALLINT,
  username VARCHAR(32) DEFAULT '',
  pv BIGINT SUM DEFAULT '0'
)
  AGGREGATE KEY(event_day, siteid, citycode, username)
  PARTITION BY RANGE(event_day)
  (
  PARTITION p201706 VALUES LESS THAN ('2017-07-01'),
  PARTITION p201707 VALUES LESS THAN ('2017-08-01'),
  PARTITION p201708 VALUES LESS THAN ('2017-09-01')
)
  DISTRIBUTED BY HASH(siteid) BUCKETS 10

PROPERTIES("replication_num" = "1");

NOTE

- When Doris creates a table, at least two copies must be specified to ensure high availability.
- You can add a rollup table to a table to improve query performance.
- By default, the Null attribute of a column in a table is **true**, which affects the query performance.
- The bucket column must be specified for the Doris table.

- View the table content.
  - **SHOW TABLES;**

```
+----------------------+
| Tables_in_example_db |
+----------------------+
| table1               |
| table2               |
+----------------------+
2 rows in set (0.01 sec)
```

  - **DESC table1;**

```
+----------+-------------+------+-------+---------+-------+
| Field    | Type        | Null | Key   | Default | Extra |
+----------+-------------+------+-------+---------+-------+
| siteid   | int(11)     | Yes  | true  | 10      |       |
| citycode | smallint(6) | Yes  | true  | N/A     |       |
| username | varchar(32) | Yes  | true  |         |       |
| pv       | bigint(20)  | Yes  | false | 0       | SUM   |
+----------+-------------+------+-------+---------+-------+
4 rows in set (0.00 sec)
```

  - **DESC table2;**

```
+-----------+-------------+------+-------+---------+-------+
| Field     | Type        | Null | Key   | Default | Extra |
+-----------+-------------+------+-------+---------+-------+
| event_day | date        | Yes  | true  | N/A     |       |
| siteid    | int(11)     | Yes  | true  | 10      |       |
| citycode  | smallint(6) | Yes  | true  | N/A     |       |
| username  | varchar(32) | Yes  | true  |         |       |
| pv        | bigint(20)  | Yes  | false | 0       | SUM   |
+-----------+-------------+------+-------+---------+-------+
5 rows in set (0.00 sec)
```

5.9.3 INSERT INTO

This section describes the basic syntax and usage of SQL statements for inserting table data to the Doris.

Basic Syntax

INSERT INTO *table_name*

[PARTITION (p1, ...)]

[WITH LABEL label]

[(column [, ...])]

[[hint [, ...]]]

{ VALUES ({ expression | DEFAULT } [, ...]) [, ...] | query }

Use Example

- Insert multiple rows of data into the **test** table at a time.

  INSERT INTO test VALUES (1, 2), (3, 4);

- Import the result of a query statement to the **test** table.

  INSERT INTO test (c1, c2) SELECT * from test2;

5.9.4 ALTER TABLE

When you modify the table structure, the methods for modifying the aggregation model and non-aggregation model are different. The methods for modifying the Key and Value columns are also different. The command parameters are as follows:

- If **AGGREGATE KEY** is specified during table creation, the model is an aggregation model. In other scenarios, the model is a non-aggregation model.
- In the table creation statement, the column following the keyword **'unique key'**, **'aggregate key'**, or **'duplicate key'** is the Key column, and the remaining column is the Value column.

Aggregation Model Example

The aggregation type of an aggregation column cannot be changed.

- Add the **new_col** column (key column) after the **col1** column.

  ALTER TABLE example_db.my_table ADD COLUMN new_col INT DEFAULT "0" AFTER col1;

- Add the **new_col** column (sum aggregation type in the Value column) after **col1**.

  ALTER TABLE example_db.my_table ADD COLUMN new_col INT SUM DEFAULT "0" AFTER col1;

- Change the type of the **col1** column to BIGINT(Key column).

  ALTER TABLE example_db.my_table MODIFY COLUMN col1 BIGINT DEFAULT "1";

- Change the type of the **col1** column to BIGINT(Value column).

  ALTER TABLE example_db.my_table MODIFY COLUMN col1 BIGINT MAX DEFAULT "1";

- Delete the **col1** column.

  ALTER TABLE example_db.my_table DROP COLUMN col1;

Example of a non-aggregation model

- Add the **new_col** column after the **col1** column (add the Key column):

  ALTER TABLE example_db.my_table ADD COLUMN new_col INT KEY DEFAULT "0" AFTER col1;

- Add the **new_col** column (Value column) after the **col1** column.

  ALTER TABLE example_db.my_table ADD COLUMN new_col INT DEFAULT "0" AFTER col1;

- Change the type of the **col1** column to BIGINT(Key column).

ALTER TABLE example_db.my_table MODIFY COLUMN col1 BIGINT KEY DEFAULT "1";

- Change the type of the **col1** column to BIGINT(Value column).

  ALTER TABLE example_db.my_table MODIFY COLUMN col1 BIGINT DEFAULT "1";

- Delete the **col1** column.

  ALTER TABLE example_db.my_table DROP COLUMN col1;

5.9.5 DROP TABLE

This section describes the basic SQL syntax and usage of Doris for deleting tables.

Basic Syntax

DROP TABLE [IF EXISTS] [db_name.]*table_name* [FORCE]**;**

Use Example

Delete the **my_table** table.

DROP TABLE IF EXISTS example_db.my_table;

5.10 Common Issues About Doris

5.10.1 What Should I Do If Occasionally Occurs During Table Creation Due to the Configuration of the SSD and HDD Data Directories?

Symptom

Error message "Failed to find enough host with storage medium and tag" was occasionally displayed during table creation.

Cause Analysis

Doris allows you to configure multiple storage paths for a BE node and specify the storage medium of the paths, such as SSDs or HDDs. Generally, only one storage path needs to be configured for each disk.

The **default_storage_medium** parameter of FE nodes is HDD by default. If only the SSD is specified in the **be.conf** file, an error is reported because no HDD medium is available during table creation. Doris does not automatically detect the actual storage medium of disks where the storage path is located. You need to explicitly indicate the storage medium in the path configuration. **.HDD** and **.SSD** are used only to identify the speed of the storage paths. They are not the actual storage medium types. If the storage medium of the BE node has no difference, you do not need to enter the suffix.

Procedure

- Change the value of **default_storage_medium** of the FE node to the correct storage medium and restart the FE node for the modification to take effect.
- Delete the explicit SSD configuration from the **be.conf** file.
- Add the properties parameter **properties {"storage_medium" = "ssd"}** when creating a table.

5.10.2 What Should I Do If RPC Timeout Error Is Reported When Stream Load Is Used?

Symptom

During data import, RPC startup timed out when the tablet writer was opened by a BE node. The following error was reported:

```
failed to open tablet writer, error=RPC call is timeout, error_text=[E1008] Reached timeout=xxx ms
```

Cause Analysis

When data is imported, a BE node opens the tablet writer, which may involve the write operation of multiple memory blocks. As a result, the RPC times out. You can adjust the RPC timeout interval to avoid the timeout error.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris** > **Configurations** > **All Configurations**.

Step 2 In the navigation pane on the left, choose **BE(Role)** > **Customization**. Add the custom parameter **tablet_writer_open_rpc_timeout_sec** to **be.conf.customized.configs**. The parameter value is the RPC timeout interval. The default value is 60. You can increase the value, for example, set this parameter to **300**.

Step 3 Click **Instance**, select all BE instances, and choose **More** > **Restart Instance** to restart the BE instances.

----End

5.10.3 What Do I Do If the Error Message "plugin not enabled" Is Displayed When the MySQL Client Is Used to Connect to the Doris Database?

Symptom

The following error message is displayed when the MySQL client is used to connect to the Doris database:

```
ERROR 2059 (HY000): Authentication plugin 'mysql_clear_password' cannot be loaded: plugin not enabled
```


Cause Analysis

The mysql_clear_password plug-in is disabled.

Procedure

- When you use the MySQL client to connect to the Doris database, run the following command with **--enable-cleartext-plugin** added:

  mysql --enable-cleartext-plugin -u*Database login user* **-p***Database login user password* **-P***Database connection port* **-h***Doris FE instance IP address*

- Run the following command on the node where the MySQL client is installed to enable the mysql_clear_password plug-in, and then reconnect to Doris:

  export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

5.10.4 How Do I Handle the FE Startup Failure?

Symptom

The FE instance failed to be started, and the **/var/log/Bigdata/doris/fe/fe.log** file kept showing the following error message:

```
wait catalog to be ready. FE type UNKNOWN
```

Cause Analysis

- The FE installation node has multiple network adapter IP addresses. The **priority_network** parameter is incorrectly set. As a result, an incorrect IP address is matched during FE startup.
- Most follower FE nodes in the cluster are not started. For example, there are three follower FE nodes and only one follower FE node is started.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris**, and click **Configurations**.

Step 2 Search for the **priority_network** parameter and set it correctly for the FE service. You can view the IP address of the network adapter bound to the FE node by checking the **CURRENT_INSTANCE_IP** variable in *FE installation directory*/**FusionInsight_Doris_*/1_*_FE/etc/ENV_VARS**.

The **priority_network** parameter is used to help the system select the correct IP address of the network adapter as the IP address of the FE or BE. You are advised to set this parameter explicitly in any case to prevent incorrect IP address selection after network adapters are added. The value of **priority_network** is in CIDR format and is used to ensure that all nodes can use the same configuration value. The parameter value consists of two parts. The first part is the IP address in dotted decimal notation, and the second part is the prefix length.

For example, 10.168.1.0/8 matches all 10.xx.xx.xx IP addresses, and 10.168.1.0/16 matches all 10.168.xx.xx IP addresses. If there are two nodes: 10.168.10.1 and 10.168.10.2, **10.168.10.0/24** can be the value of **priority_network**.

Step 3 Click **Instance**, select the follower FE to be started, and click **Start Instance**. For example, if there are three followers and only one follower is started, you need to

start at least one FE so that a master node can be elected from the FE election group to provide services.

Step 4 If the FE still fails to be started, contact O&M engineers to rectify the fault.

----End

5.10.5 How Do I Handle the Startup Failure Due to Incorrect IP Address Matching for the BE Instance?

Symptom

The BE instance failed to be started and the following error message was displayed:

```
backend ip saved in master does not equal to backend local ipx.x.x.x vs. x.x.x.x
```

Cause Analysis

The BE installation node has multiple network adapter IP addresses. The **priority_network** parameter is incorrectly set. As a result, an incorrect IP address is matched during BE startup.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris**, and click **Configurations**.

Step 2 Search for the **priority_network** parameter and set it correctly for the BE service. You can view the IP address of the network adapter bound to the BE node by checking the **CURRENT_INSTANCE_IP** variable in *BE installation directory*/**FusionInsight_Doris_*/1_*_BE/etc/ENV_VARS**.

The **priority_network** parameter is used to help the system select the correct IP address of the network adapter as the IP address of the FE or BE. You are advised to set this parameter explicitly in any case to prevent incorrect IP address selection after network adapters are added. The value of **priority_network** is in CIDR format and is used to ensure that all nodes can use the same configuration value. The parameter value consists of two parts. The first part is the IP address in dotted decimal notation, and the second part is the prefix length.

For example, 10.168.1.0/8 matches all 10.xx.xx.xx IP addresses, and 10.168.1.0/16 matches all 10.168.xx.xx IP addresses. If there are two nodes: 10.168.10.1 and 10.168.10.2, **10.168.10.0/24** can be the value of **priority_network**.

----End

5.10.6 What Should I Do If Error Message "Read timed out" Is Displayed When the MySQL Client Connects to the Doris?

Symptom

An error was reported when the MySQL client is connected to Doris.

```
java.net.SocketTimeoutException: Read timed out
```


Cause Analysis

The Doris server responds slowly.

Procedure

When you use the MySQL client to connect to the Doris database, run the following command with **connect_timeout** added (default value: 10 seconds):

mysql -u*Database login user* **-p***Database login user password* **-P***Database connection port* **-h***IP address of Doris FE instance* **--connect_timeout=***120*

5.10.7 What Should I Do If an Error Is Reported When the BE Runs a Data Import or Query Task?

Symptom

When data is imported or queried, the following error message is displayed:

```
Not connected to 192.168.100.1:8060 yet, server_id=384
```

Cause Analysis

- The BE node where the task is running breaks down.
- RPC congestion or other errors occur.

Procedure

- If the BE node where the task is running breaks down, check the breakdown cause and rectify the fault.
- If a large amount of unsent data on the RPC source exceeds the threshold, set the following parameters:
  - **brpc_socket_max_unwritten_bytes**: specifies the threshold of the amount of data that is not sent. The default value is 1 GB. If the amount of data that is not sent exceeds the threshold value, the OVERCROWDED error is reported. You need to increase the value.
  - **tablet_writer_ignore_eovercrowded**: specifies whether to ignore OVERCROWDED error that occurs during data import. The default value is **false**. This parameter is used to prevent import failures and improve stability.
  - **max_body_size**: specifies the maximum packet size allowed. The default value is 3 GB. If the query contains data of the string or bitmap type, you can modify this parameter to avoid this problem.

5.10.8 What Should I Do If a Timeout Error Is Reported When Broker Load Imports Data?

Symptom

The following error message was displayed when Broker Load was used to import data:


```
org.apache.thrift.transport.TTransportException: java.net.SocketException: Broken pipe
```

Cause Analysis

When data is imported from an external storage device (for example, HDFS), file directory query times out because there are too many files in the directory.

Procedure

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** and click **Configurations** > **All Configurations**. Select **FE (Role)** > **Customization**, and add the custom parameter **broker_timeout_ms**. The default value is 10 seconds. You need to increase the value of this parameter, for example, to **1000**, and restart the FE instance whose configuration has expired.

5.10.9 What Should I Do If an Error Message Is Displayed When Broker Load Is Used to Import Data?

Symptom

When Broker Load is used to import data, the error message "failed to send batch" or "TabletWriter add batch with unknown id" is displayed.

Cause Analysis

The task execution times out due to a large number of concurrent requests or a large amount of data.

Procedure

Step 1 Log in to the MySQL client and run the following command to increase the value of **query_timeout**. The default value is 300 seconds.

SET GLOBAL query_timeout = *xxx*;

Step 2 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** and click **Configurations** > **All Configurations**. Select **BE (Role)** > **Customization**, and add the custom parameter **streaming_load_rpc_max_alive_time_sec**. The default value is 1200 seconds. You need to increase the value of this parameter and restart the BE instance whose configuration has expired.

----End

5.11 Doris Troubleshooting

5.11.1 What Should I Do If a Query Is Performed on the BE Node Where Some Copies Are Lost or Damaged and an Error Is Reported?

Symptom

If multiple copies were lost from the disk, for example, **mv** was renamed, the kernel could not detect the loss of the copies. If the query request runs on the BE node where the copies are lost, the following error is reported:

Procedure

Step 1 Log in to the node where MySQL is installed and connect to the Doris database.

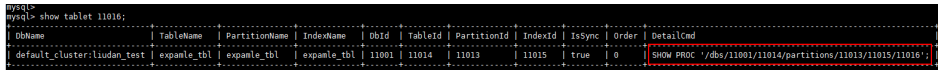
Step 2 Call **check_tablet_segment_lost** of the BE node to request automatic recovery of lost copies.

curl -X POST http://*192.168.67.78:29986***/api/check_tablet_segment_lost?repair=true** (

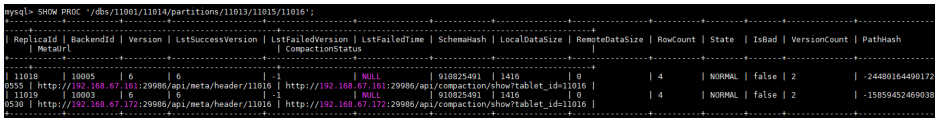
In the preceding command, *192.168.67.78* indicates the IP address of the abnormal BE node, and *29986* indicates the service port of the HTTP server of the BE node. You can search for **webserver_port** on the Doris configuration page of Manager to view the port.

Step 3 Run the following command to obtain DetailCmd:

show tablet *tabletId*



Step 4 Run DetailCmd. If the copies of the abnormal node have been removed, run the query again.



----End

5.11.2 How Do I Restore the FE Service from a Fault?

Symptom

The FE service failed to start bdbje, data could not be synchronized between FE nodes, metadata could not be written, or no master node was available. To restore the FE service, start a new master node based on the metadata in **meta_dir**, and then add FE nodes one by one.

Procedure

Step 1 Stop all FE processes and stop all service access to prevent unexpected problems caused by external access during metadata restoration.

Step 2 Search for the metadata on all FE instance nodes and locate the latest FE node as the master node to be restored.

1. Log in to the FE background node and check the value of **meta_dir** in the **${BIGDATA_HOME}/FusionInsight_Doris_x.x.x/x_x_FE/etc/fe.conf** file. The value is the metadata storage directory.
2. Search for the metadata storage directories of all FE nodes and check the **image/image.***xxxx* files in the directories. A larger value of **image.***xxxx* indicates a newer metadata. Locate the latest FE node and use it as the first FE to be restored, that is, the master FE.
3. Back up the metadata storage directories of all FEs.

   For example, if the metadata storage directory is **/srv/BigData/doris_fe/doris-meta**, run the following command:

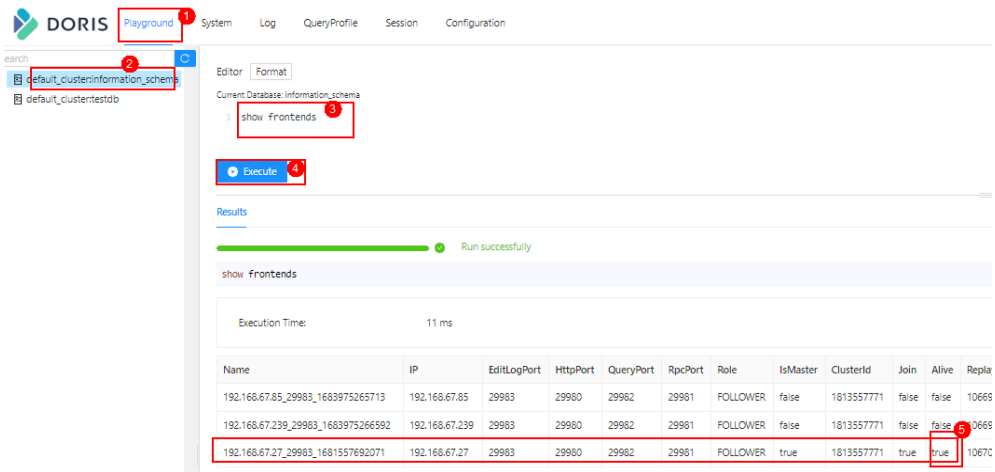
   cp -r /srv/BigData/doris_fe/doris-meta /srv/BigData/doris_fe/doris-meta.bak

Step 3 Go to **Step 2** and locate the node where the FE node with the latest metadata is deployed (that is, the master node) and add **metadata_failure_recovery=true** to **${BIGDATA_HOME}/FusionInsight_Doris_x.x.x/x_x_FE/etc/fe.conf**. If the **${BIGDATA_HOME}/FusionInsight_Doris_x.x.x/x_x_FE_UPDATE** directory exists, add the configuration to **fe.conf** in **x_x_FE_UPDATE**.

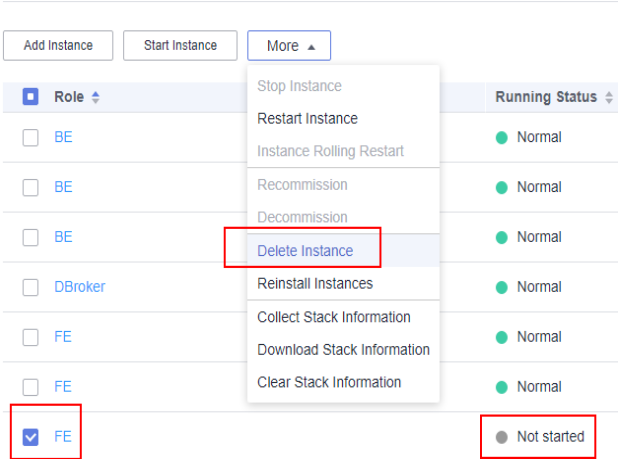
Step 4 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Instances**, select the FE node whose configuration is modified in **Step 3**, and choose **More** > **Restart Instance** to restart the FE instance. Other instances are still stopped.

Step 5 Check the status of the FE instance after it is started. After the FE instance is started, enter **http://***192.168.67.27*:*29980* in the address box of the browser to connect to the FE instance.

Log in to the FE web UI, click **Playground**, select **default_cluster:information_schema**, enter the **show frontends** command in the command box on the right, and click **Execute**. If the value in the **Alive** column of the current FE instance in the **Results** list is **true**, the FE is restored.

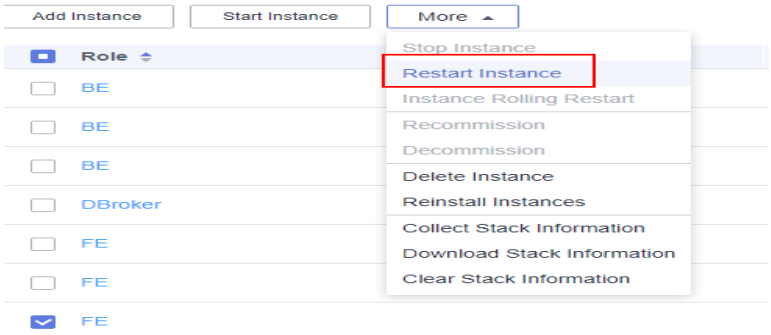


Step 6 On FusionInsight Manager, choose **Cluster** > **Services** > **Instances**, select the FE instance that is not a Master node and is not started, and choose **More** > **Delete Instance**.



Step 7 After the FE instance is deleted, click **Add Instance** to add the FE instance deleted in **Step 6**.

Step 8 Select the instance whose configuration has expired, choose **More** > **Restart Instance** to restart the FE instance whose configuration has expired, and delete the **metadata_failure_recovery** parameter added to the **fe.conf** file on the node where the FE instance is deployed.

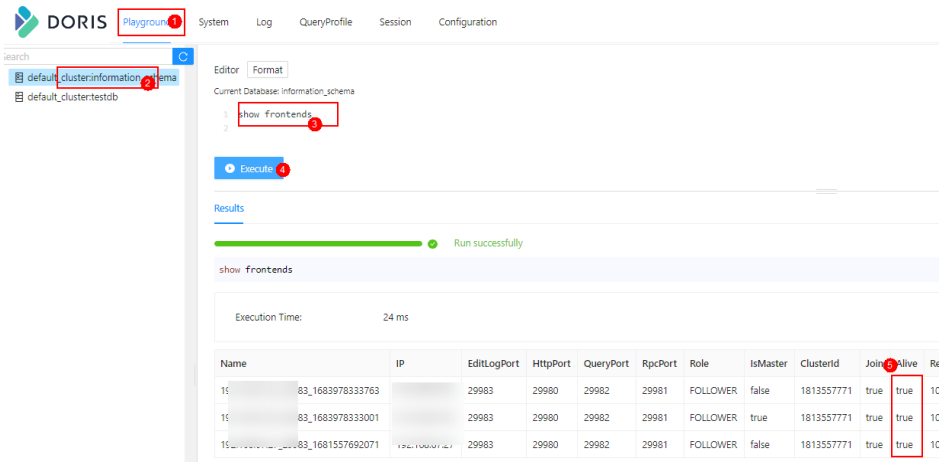


Step 9 Check whether the cluster is running properly. On the FE web UI, run the following command to check whether the FE, BE, and DBroker processes are healthy and in the same cluster. If the value of **Alive** for all instances in the **Results** list is **true**, the processes are healthy.

For example, the following commands are executed in **default_cluster:information_schema** of **Playground** on the Doris web UI:

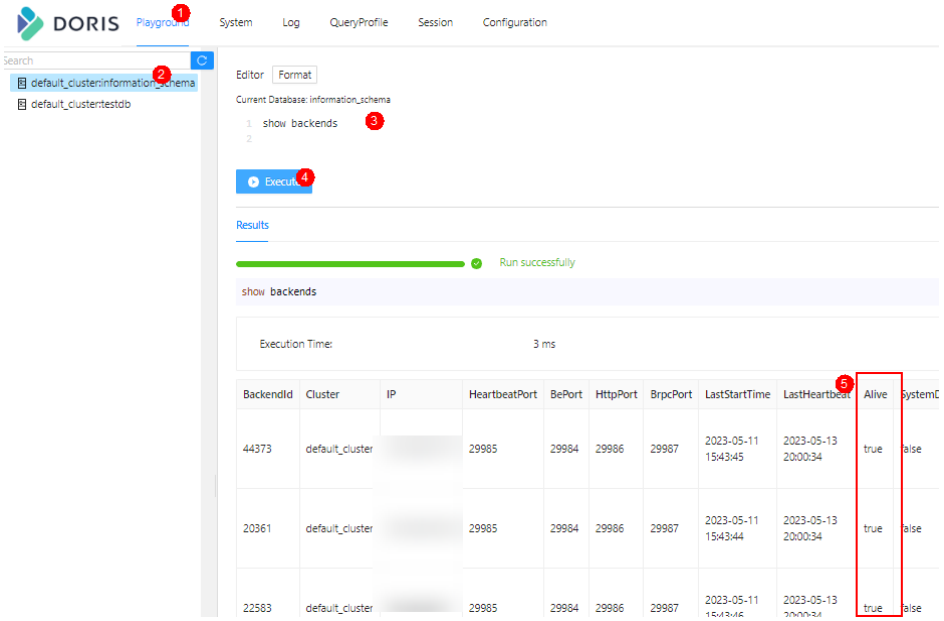
- Run the following command to check whether all FE processes are healthy:

  show frontends;



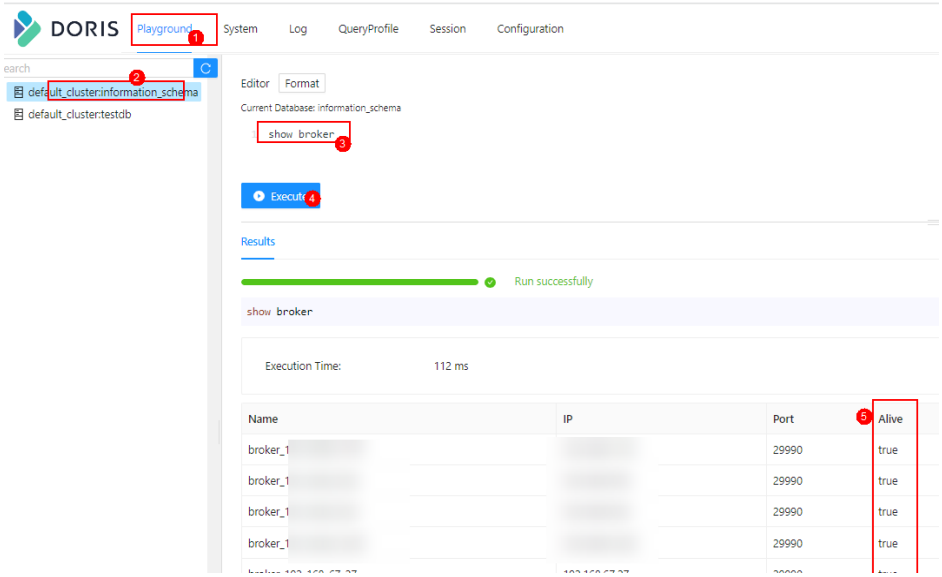
- Run the following command to check whether all BE processes are healthy:

  show backends;



- Run the following command to check whether all DBroker processes are healthy:

  show broker;



----End

5.11.3 What Should I Do If the Data Volume of a Broker Load Import Task Exceeds the Threshold?

Symptom

The following error message was displayed when Broker Load was used to import data:

```
Scan bytes per broker scanner exceed limit:xxx
```

Cause Analysis

The maximum data volume of a single import task processed by the BE process is 3 GB. You can adjust the import parameters of Broker Load for large files.

Procedure

Modify the maximum scan amount and maximum concurrent number of tasks of a single BE node according to the current number of BE instances and the size of the files to be imported. Perform the following steps to modify the parameters:

1. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris**. On the dashboard page, view the IP address of leader host to determine the node where the active FE node is deployed.
2. Click **Instance**, and click the BE instance whose IP address was viewed in **1**. Click **Configurations** > **All Configurations**, select **BE (Role)** > **Customization**, and add the following parameters:
   - **max_broker_concurrency**: number of BE nodes
   - **max_bytes_per_broker_scanner**: size of the file to be imported/number of BE nodes

NOTE

The configuration items take effect only when they are modified in the **fe.conf** file of the leader FE.

3. Click **Save** to save the configuration and restart the instance whose configuration has expired.

6 Using Flink

6.1 Flink Job Engine

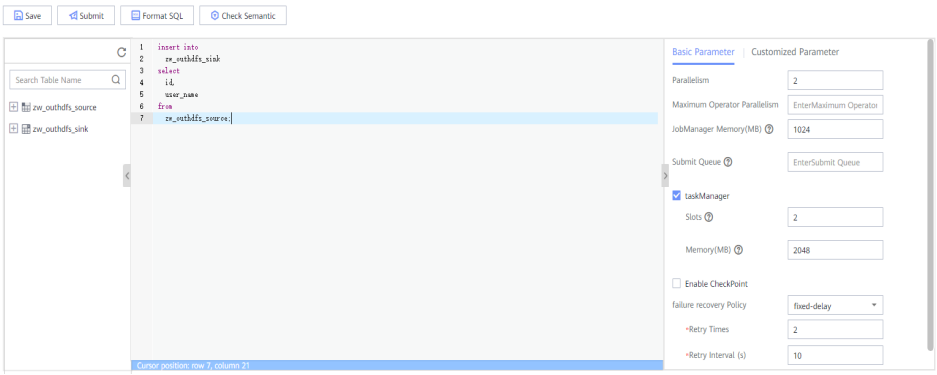
Flink web UI provides a web-based visual development platform. You only need to compile SQL statements to develop jobs, slashing the job development threshold. In addition, the exposure of platform capabilities allows service personnel to compile SQL statements for job development to quickly respond to requirements, greatly reducing the Flink job development workload.

Flink Web UI Features

The Flink web UI has the following features:

- Enterprise-class visual O&M: GUI-based O&M management, job monitoring, and standardization of Flink SQL statements for job development.

  Figure 6-1 Operations management

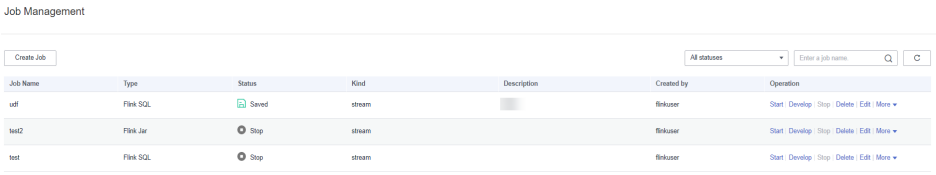
- Quick cluster connection: After configuring the client and user credential key file, you can quickly access a cluster using the cluster connection function.
- Quick data connection: You can access a component by configuring the data connection function. If **Data Connection Type** is set to **HDFS**, you need to create a cluster connection. If **Authentication Mode** is set to **KERBEROS** for other data connection types, you need to create a cluster connection. If **Authentication Mode** is set to **SIMPLE**, you do not need to create a cluster connection.

NOTE

If **Data Connection Type** is set to **Kafka**, **Authentication Type** cannot be set to **KERBEROS**.

- Visual development platform: The input/output mapping table can be customized to meet the requirements of different input sources and output destinations.
- Easy to use GUI-based job management

Figure 6-2 Job management



Key Web UI Capabilities

Table 6-1 shows the key capabilities provided by Flink web UI.

Table 6-1 Key web UI capabilities

Item	Description
Batch-Stream convergence	• Batch jobs and stream jobs can be processed with a unified set of Flink SQL statements.
Flink SQL kernel capabilities	• Flink SQL supports customized window size, stream compute within 24 hours, and batch processing beyond 24 hours. • Flink SQL supports reading data from Kafka and HDFS, writing data to Kafka and HDFS. • A job can define multiple Flink SQL statements, and multiple metrics can be combined into one computing job. If a job contains same primary keys as well as same inputs and outputs, the job supports the computing of multiple windows. • The AVG, SUM, COUNT, MAX, and MIN statistical methods are supported.
Flink SQL functions on the console	• Cluster connection management allows you to configure clusters where services such as Kafka and HDFS reside. • Data connection management allows you to configure services such as Kafka and HDFS. • Data table management allows you to define data tables accessed by SQL statements and generate DDL statements. • Flink SQL job definition allows you to use SQL statements to verify, parse, optimize, convert a job into a Flink job, and submit the job.

Item	Description
Flink job visual management	• Stream jobs and batch jobs can be defined in a visual manner. • Job resources, fault recovery policies, and checkpoint policies can be configured in a visual manner. • Status monitoring of stream and batch jobs are supported. • The Flink job O&M is enhanced, including redirection of the native monitoring page.
Performance and reliability	• Stream processing supports 24-hour window aggregation computing and millisecond-level performance. • Batch processing supports 90-day window aggregation computing, which can be completed in minutes. • Invalid data of stream processing and batch processing can be filtered out. • When HDFS data is read, the data can be filtered based on the calculation period in advance. • If the job definition platform is faulty or the service is degraded, jobs cannot be redefined, but the computing of existing jobs is not affected. • The automatic restart mechanism is provided for job failures. You can configure restart policies.

Flink Web UI Application Process

The Flink web UI application process is shown as follows:

Figure 6-3 Flink web UI application process

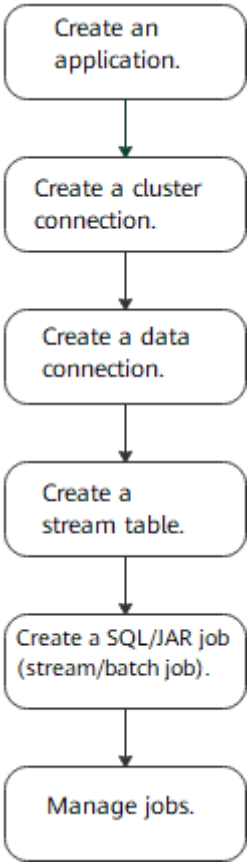


Table 6-2 Description of the Flink web UI application process

Step	Description	Reference
Creating an application	Applications can be used to isolate different upper-layer services.	**Creating a FlinkServer Application**
Creating a cluster connection	Different clusters can be accessed by configuring the cluster connection.	**Creating a FlinkServer Cluster Connection**
Creating a Data Connection	Through data connections, you can access different data services, including HDFS and Kafka.	**Creating a FlinkServer Data Connection**
Creating a stream table	Data tables can be used to define basic attributes and parameters of source tables, dimension tables, and output tables.	**Creating a FlinkServer Stream Table Source**
Creating a SQL/JAR job (stream/batch job)	APIs can be used to define Flink jobs, including Flink SQL and Flink Jar jobs.	**Creating a FlinkServer Job**

Step	Description	Reference
Managing jobs	A created job can be managed, including starting, developing, stopping, deleting, and editing the job.	Creating a FlinkServer Job

6.2 Flink User Permission Management

6.2.1 Flink Security Authentication

Flink Authentication and Encryption

- In a Flink cluster, all components support authentication.
  - The kerberos authentication is supported between Flink cluster components and external components, such as YARN, HDFS, and ZooKeeper.
  - The security cookie authentication between Flink cluster components, for example, Flink client and JobManager, JobManager and TaskManager, and TaskManager and TaskManager, are supported.
- SSL encrypted transmission is supported by components of a Flink cluster and between components in a cluster, such as Flink client and JobManager, JobManager and TaskManager, as well as TaskManager and TaskManager.

ACL Control

In HA mode, ACL control is supported.

In HA mode of Flink, ZooKeeper can be used to manage clusters and discover services. Zookeeper supports SASL ACL control. Only users who have passed the SASL (Kerberos) authentication have the permission to operate files on ZooKeeper. To enable SASL ACL control, perform following configurations in the Flink configuration file.

```
high-availability.zookeeper.client.acl: creator
zookeeper.sasl.disable: false
```

For details about configuration items, see **HA**.

Web Security

Security of the Flink web UI is hardened, the whitelist filtering feature is provided, the Flink web UI is accessible only through a YARN proxy, and security headers are enhanced. In Flink clusters, listening ports of components can be configured.

- Encoding rules
  - Description: The same encoding mode is used on the web service client and server to prevent garbled characters and to implement input verification.
  - Security hardening: Response messages of web servers are encoded using UTF-8.

- Whitelist-based filter of IP addresses
  - Note: IP filter must be added to the web server to filter unauthorized requests from the source IP address and prevent unauthorized login.
  - Security hardening: Add **jobmanager.web.allow-access-address** to enable the IP filter. By default, only YARN users are supported.

    NOTE

    After the client is installed, you need to add the IP address of the client node to the **jobmanager.web.allow-access-address** configuration item.
- Preventing sending the absolute paths to the client
  - Note: If an absolute path is sent to a client, the directory structure of the server is exposed, increasing the risk that attackers know and attack the system.
  - Security hardening: If the Flink configuration file contains a parameter starting with a slash (/), the first-level directory is deleted.
- Same-origin policy
  - If two URL protocols have same hosts and ports, they are of the same origin. Protocols of different origins cannot access each other, unless the source of the visitor is specified on the host of the service to be visited.
  - Security hardening: The default value of the header of the response header **Access-Control-Allow-Origin** is the IP address of ResourceManager on Yarn clusters. If the IP address is not from Yarn, mutual access is not allowed.
- Preventing sensitive information disclosure
  - Web pages containing sensitive data must not be cached, to avoid leakage of sensitive information or data crosstalk among users who visit the internet through the proxy server.
  - Security hardening: Add **Cache-control**, **Pragma**, **Expires** security header. The default value is **Cache-Control: no-store**, **Pragma: no-cache**, and **Expires: 0**. The security hardening stops contents interacted between Flink and web server from being cached.
- Anti-hijacking
  - Since hotlinking and clickjacking use framing technologies, security hardening is required to prevent attacks.
  - Security hardening: Add **X-Frame-Options** security header to specify whether the browser will load the pages from **iframe**, **frame** or **object**. The default value is **X-Frame-Options: DENY**, indicating that no pages can be nested to **iframe**, **frame** or **object**.
- Logging calls of the Web Service APIs
  - Calls of the **Flink webmonitor restful** APIs are logged.
  - The **jobmanager.web.accesslog.enable** can be added in the **access log**. The default value is **true**. Logs are stored in a separate **webaccess.log** file.
- Cross-site request forgery prevention
  - In **Browser/Server** applications, CSRF must be prevented for operations involving server data modification, such as adding, modifying, and deleting. The CSRF forces end users to execute non-intended operations on the current web application.

  - Security hardening: Only two post APIs, one delete API, and get interfaces are reserve for modification requests. All other APIs are deleted.
- Troubleshooting:
  - When the application is abnormal, exception information is filtered, logged, and returned to the client.
  - Security hardening: A default error message page to filter information and log detailed error information. Four configuration parameters are added to ensure that the error page is switched to a specified URL provided by FusionInsight, preventing exposure of unnecessary information.

Table 6-3 Parameters

Parameter	Description	Mandatory
jobmanager.web.403-redirect-url	Web page access error 403. If 403 error occurs, the specified page will be redirected to.	Yes
jobmanager.web.404-redirect-url	Web page access error 404. If 404 error occurs, the specified page will be redirected to.	Yes
jobmanager.web.415-redirect-url	Web page access error 415. If 415 error occurs, the specified page will be redirected to.	Yes
jobmanager.web.500-redirect-url	Web page access error 500. If 500 error occurs, the specified page will be redirected to.	Yes

- HTML5 security
  - HTML5 is a next generation web development specification that provides new functions and extend the labels for developers. These new labels and functions increase the attack surface and pose attack risks (such as cross-domain resource sharing, client storage, WebWorker, WebRTC, and WebSocket).
  - Security hardening: Add the **Access-Control-Allow-Origin** parameter. For example, if you want to enable the cross-domain resource sharing, configure the **Access-Control-Allow-Origin** parameter of the HTTP response header.

    NOTE

    Flink does not involve security risks of functions such as storage on the client, WebWorker, WebRTC, and WebSocket.

Security Statement

- All security functions of Flink are provided by the open source community or self-developed. Security features that need to be configured by users, such as authentication and SSL encrypted transmission, may affect performance.

- As a big data computing and analysis platform, Flink does not detect sensitive information. Therefore, you need to ensure that the input data is not sensitive.
- You can evaluate whether configurations are secure as required.
- For any security-related problems, contact O&M support.

6.2.2 Flink User Permissions

To access and use the Flink web UI to perform service operations, you need to assign FlinkServer-related permissions to users. User **admin** on FusionInsight Manager does not have FlinkServer service operation permissions.

Applications (tenants) in FlinkServer are the maximum management scope, including cluster connection management, data connection management, application management, stream table management, and job management.

There are three types of resource permissions for FlinkServer, as shown in **Table 6-4**.

Table 6-4 FlinkServer resource permissions

Name	Description	Remarks
FlinkServer administrator permission	Users who have the permission can edit and view all applications.	This is the highest-level permission of FlinkServer. If you have the FlinkServer administrator permission, you have the permission on all applications by default.
Application edit permission	Users who have the permission can create, edit, and delete cluster connections and data connections. They can also create stream tables as well as create and run jobs.	In addition, users who have the permission can view current applications.
Application view permission	Users who have the permission can view applications.	-

6.2.3 Creating a FlinkServer Role

Create and configure a FlinkServer role on Manager as an MRS cluster administrator. A FlinkServer role can be configured with the FlinkServer administrator permission and the permissions to edit and view applications.

You need to set permissions for the specified user in FlinkServer so that they can update, query, and delete data.

Prerequisites

The cluster administrator has planned permissions based on service requirements.

Procedure

Step 1 Log in to Manager.

Step 2 Choose **System** > **Permission** > **Role**.

Step 3 On the displayed page, click **Create Role** and specify **Role Name** and **Description**.

Step 4 Set **Configure Resource Permission**.

FlinkServer permissions are as follows:

- **FlinkServer Admin Privilege**: highest-level permission. Users with the permission can perform service operations on all FlinkServer applications.
- **FlinkServer Application**: Users can set **application view** and **applications management** permissions on applications.

Table 6-5 Setting a role

Scenario	Role Authorization
Setting the FlinkServer administrator permission	In **Configure Resource Permission**, choose *Name of the desired cluster* > **Flink** and select **FlinkServer Admin Privilege**.
Setting FlinkServer application permissions	In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Flink** > **FlinkServer Application**. In the **Permission** column, select **application view** or **applications management**.

Step 5 Click **OK**. Return to role management page.

Step 6 (Optional) Create a user with FlinkServer-related permissions.

After the FlinkServer role is created, create a FlinkServer user and bind the user to the configured FlinkServer role and user group.

----End

6.2.4 Configuring Security Authentication for Interconnecting with Kafka

Sample project data of Flink is stored in Kafka. A user with Kafka permission can send data to Kafka and receive data from it.

Step 1 Ensure that clusters, including HDFS, Yarn, Flink, and Kafka are installed.

Step 2 Create a topic.

- Run Linux command line to create a topic. Before running commands, ensure that the kinit command, for example, **kinit flinkuser**, is run for authentication.

  NOTE

  To create a Flink user, you need to have the permission to create Kafka topics.

The format of the command is shown as follows, in which **{zkQuorum}** indicates ZooKeeper cluster information and the format is *IP.port*, and **{Topic}** indicates the topic name.

bin/kafka-topics.sh --create --zookeeper {zkQuorum}/kafka --replication-factor 1 --partitions 5 --topic {Topic}

Assume the topic name is **topic 1**. The command for creating this topic is displayed as follows:

```
/opt/client/Kafka/kafka/bin/kafka-topics.sh --create --zookeeper 10.96.101.32:2181,10.96.101.251:2181,10.96.101.177:2181,10.91.8.160:2181/kafka --replication-factor 1 --partitions 5 --topic topic1
```

NOTE

The ZooKeeper cluster information is as follows:

- Service IP address of the ZooKeeper quorumpeer instance:

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ZooKeeper**. On the page that is displayed, click the **Instance** tab and view the service IP addresses of all nodes where the quorumpeer instances reside.

- Port number of the ZooKeeper client:

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ZooKeeper**. On the page that is displayed, click the **Configurations** tab. On this tab page, view the value of **clientPort**.

- Configure the permission of the topic on the server.

  Set the **allow.everyone.if.no.acl.found** parameter of Kafka Broker to **true**.

Step 3 Perform the security authentication.

The Kerberos authentication, SSL encryption authentication, or Kerberos + SSL authentication mode can be used.

- **Kerberos authentication**
  - Client configuration

    In the Flink configuration file **flink-conf.yaml**, add configurations about Kerberos authentication. For example, add **KafkaClient** in **contexts** as follows:

    ```
    security.kerberos.login.keytab: /home/demo/keytab/flinkuser.keytab
    security.kerberos.login.principal: flinkuser
    security.kerberos.login.contexts: Client,KafkaClient
    security.kerberos.login.use-ticket-cache: false
    ```

  - Running parameter

    Running parameters about the **SASL_PLAINTEXT** protocol are as follows:

    --topic topic1 --bootstrap.servers 10.96.101.32:21007 --security.protocol SASL_PLAINTEXT --sasl.kerberos.service.name kafka --kerberos.domain.name hadoop.*System domain name***.com //10.96.101.32:21007** indicates the IP address and port of the Kafka server.

- **SSL encryption**
  - Configure the server.

    Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka** > **Configurations**, and set **Type** to **All**. Search for **ssl.mode.enable** and set it to **true**.

  - Configure the client.

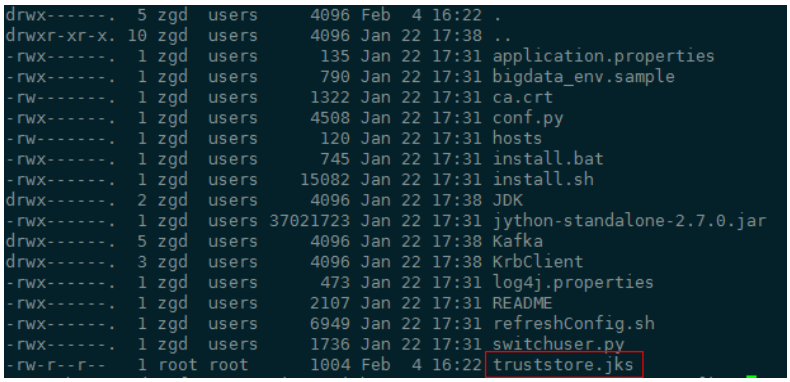
    i. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, click **More**, and choose **Download Client** to download the Kafka client.

ii. Use the **ca.crt** certificate file in the client root directory to generate the **truststore** file for the client.

Run the following command:

```
keytool -noprompt -import -alias myservercert -file ca.crt -keystore truststore.jks
```

The command execution result is similar to the following:



iii. Run parameters.

The value of **ssl.truststore.password** must be the same as the password you entered when creating **truststore**. Run the following command to run parameters. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

```
--topic topic1 --bootstrap.servers 10.96.101.32:9093 --security.protocol SSL --ssl.truststore.location /home/zgd/software/FusionInsight_Kafka_ClientConfig/truststore.jks --ssl.truststore.password XXX
```

- **Kerberos+SSL encryption**

  After completing preceding configurations of the client and server of Kerberos and SSL, modify the port number and protocol type in running parameters to enable the Kerberos+SSL encryption mode.

  ```
  --topic topic1 --bootstrap.servers 10.96.101.32:21009 --security.protocol SASL_SSL --sasl.kerberos.service.name kafka --ssl.truststore.location --kerberos.domain.name hadoop.System domain name.com /home/zgd/software/FusionInsight_Kafka_ClientConfig/truststore.jks --ssl.truststore.password XXX
  ```

----End

6.2.5 Configuring Flink Authentication and Encryption

Security Authentication

Flink supports the following authentication modes:

- Kerberos authentication: It is used between the Flink Yarn client and Yarn ResourceManager, JobManager and ZooKeeper, JobManager and HDFS, TaskManager and HDFS, Kafka and TaskManager, as well as TaskManager and ZooKeeper.
- Security cookie authentication: Security cookie authentication is used between Flink Yarn client and JobManager, JobManager and TaskManager, as well as TaskManager and TaskManager.
- Internal authentication of Yarn: The Internal authentication mechanism of Yarn is used between Yarn ResourceManager and ApplicationMaster (AM).

NOTE

- Flink JobManager and Yarn ApplicationMaster are in the same process.
- If Kerberos authentication is enabled for the user's cluster, Kerberos authentication is required.

Table 6-6 Authentication modes

Authentication Mode	Description	Configuration Method

Kerberos authentication	Currently, only keytab authentication mode is supported.	1. Download the user keytab file from FusionInsight Manager and save the keytab file to a folder on the host where the Flink client is located.
		2. Configure the following parameters in the **flink-conf.yaml** file:
		a. Keytab path
		`security.kerberos.login.keytab: /home/flinkuser/keytab/abc222.keytab`
		NOTE
		/home/flinkuser/keytab/abc222.keytab indicates the user directory.
		b. Principal name
		`security.kerberos.login.principal: abc222`
		c. In HA mode, if ZooKeeper is configured, the Kerberos authentication configuration items must be configured as follows:
		`zookeeper.sasl.disable: false`
		`security.kerberos.login.contexts: Client`
		d. If you want to perform Kerberos authentication between Kafka client and Kafka broker, set the value as follows:
		`security.kerberos.login.contexts: Client,KafkaClient`

Authentication Mode	Description	Configuration Method
Security cookie authentication	-	1. In the **bin** directory of the Flink client, call the **generate_keystore.sh** script to generate **Security cookie**, **flink.keystore**, and **flink.truststore**. Run the **sh generate_keystore.sh** command and enter the user-defined password. The password cannot contain **#**. **NOTE** After the script is executed, the **flink.keystore** and **flink.truststore** files are generated in the **conf** directory on the Flink client. In the **flink-conf.yaml** file, default values are specified for following parameters: • Set **security.ssl.keystore** to the absolute path of the **flink.keystore** file. • Set **security.ssl.truststore** to the absolute path of the **flink.truststore** file. • Set **security.cookie** to a random password automatically generated by the **generate_keystore.sh** script. • By default, **security.ssl.encrypt.enabled: false** is set in the **flink-conf.yaml** file by default. The **generate_keystore.sh** script sets **security.ssl.key-password**, **security.ssl.keystore-password**, and **security.ssl.truststore-password** to the password entered when the **generate_keystore.sh** script is called. • If ciphertext is required and **security.ssl.encrypt.enabled** is **true** in the **flink-conf.yaml** file, the **generate_keystore.sh** script does not set **security.ssl.key-password**, **security.ssl.keystore-password**, and **security.ssl.truststore-password**. To obtain the values, use the Manager plaintext encryption API by running the following command: **curl -k -i -u** *Username*:*Password* **-X POST -HContent-type:application/json -d '{"plainText":"***Password***"}' 'https://***x.x.x.x***:28443/web/api/v2/tools/encrypt'** In the preceding command, *Username*:*Password* indicates the user name and password for logging in to the system. The password of **"plainText"** indicates the one used to call the **generate_keystore.sh** script. *x.x.x.x* indicates the floating IP address of Manager. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage. 2. Check whether **Security Cookie** is enabled. That is, check **security.enable: true** in the **flink-conf.yaml** file and check whether **security cookie** is configured. An example is as follows: `security.cookie: ae70acc9-9795-4c48-ad35-8b5adc8071744f605d1d-2726-432e-88ae-dd39bfec40a9`

Authentication Mode	Description	Configuration Method
		NOTE The validity period of the SSL certificate obtained by using the **generate_keystore.sh** script preset on the MRS client is 5 years. To disable the default SSL authentication mode, set **security.ssl.enabled** to **false** in the **flink-conf.yaml** file and comment out **security.ssl.key-password**, **security.ssl.keystore-password**, **security.ssl.keystore**, **security.ssl.truststore-password**, and **security.ssl.trustore**.
Internal authentication of Yarn	This authentication mode does not need to be configured by the user.	-

NOTE

One Flink cluster supports only one user. One user can create multiple Flink clusters.

Encrypted Transmission

Flink supports three encrypted transmission modes:

- Encrypted transmission inside Yarn: It is used between the Flink Yarn client and Yarn ResourceManager, as well as Yarn ResourceManager and JobManager.
- SSL transmission: SSL transmission is used between Flink Yarn client and JobManager, JobManager and TaskManager, as well as TaskManager and TaskManager.
- Encrypted transmission inside Hadoop: The internal encrypted transmission mode of Hadoop used between JobManager and HDFS, TaskManager and HDFS, JobManager and ZooKeeper, as well as TaskManager and ZooKeeper.

NOTE

Configuration about SSL encrypted transmission is mandatory while configuration about encryption of Yarn and Hadoop is not required.

To configure SSL encrypted transmission, configure the following parameters in the **flink-conf.yaml** file on the client:

1. Turn on the SSL switch and set SSL encryption algorithms. **Table 6-7** describes the parameters. Set the parameters based on your need.

Table 6-7 Parameter description

Parameter	Example Value	Description
security.ssl.enabled	true	Enable SSL.
akka.ssl.enabled	true	Enable Akka SSL.
blob.service.ssl.enabled	true	Enable SSL for the Blob channel.
taskmanager.data.ssl.enabled	true	Enable SSL transmissions between TaskManagers.
security.ssl.algorithms	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Configure the SSL encryption algorithm.

NOTE

Enabling SSL for data transmission between TaskManagers may pose great impact on the system performance.

2. In the **bin** directory of the Flink client, run the ***sh generate_keystore.sh*** *<Password>* command. The configuration items in **Table 6-8** are set by default. You can change the configurations. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

Table 6-8 Parameter description

Parameter	Example Value	Description
security.ssl.keystore	${path}/flink.keystore	Path for storing the **keystore**. **flink.keystore** indicates the name of the **keystore** file generated by the **generate_keystore.sh*** tool.
security.ssl.keystore-password	-	A user-defined password of **keystore**.
security.ssl.key-password	-	A user-defined password of the SSL key.

Parameter	Example Value	Description
security.ssl.truststore	${path}/flink.truststore	Path for storing the **truststore**. **flink.truststore** indicates the name of the **truststore** file generated by the **generate_keystore.sh*** tool.
security.ssl.truststore-password	-	A user-defined password of **truststore**.

NOTE

The **path** directory is a user-defined directory for storing configuration files of the SSL keystore and truststore.

3. Configure the paths for the client to access the **keystore** or **truststore** file.
   - Relative path (recommended)

     Perform the following steps to set the file paths of **flink.keystore** and **flink.truststore** to relative paths and ensure that the directory where the Flink client command is executed can directly access the relative paths.

     i. In the **conf/** directory of the Flink client, create a directory, for example, **ssl**.

     cd / *Flink client directory*/**Flink/flink/conf/**

     mkdir ssl

     ii. Move the **flink.keystore** and **flink.truststore** files to the new paths.

     mv flink.keystore ssl/

     mv flink.truststore ssl/

     iii. Change the values of the following parameters to relative paths in the **flink-conf.yaml** file:

     vi / *Flink client directory*/**Flink/flink/conf/flink-conf.yaml**

     ```
     security.ssl.keystore: ssl/flink.keystore
     security.ssl.truststore: ssl/flink.truststore
     ```

   - Absolute path

     After the **generate_keystore.sh** script is executed, the **flink.keystore** and **flink.truststore** file paths are automatically set to absolute paths in the **flink-conf.yaml** file by default. In this case, you need to place the **flink.keystore** and **flink.truststore** files in the **conf** directory to the absolute paths of the Flink client and each Yarn node, respectively.

6.3 Using the Flink Client

This section describes how to use Flink to run wordcount jobs.

Prerequisites

- Flink has been installed in an MRS cluster.

- The cluster runs properly and the client has been correctly installed, for example, in the **/opt/hadoopclient** directory. The client directory in the following operations is only an example. Change it to the actual installation directory.

Procedure

Step 1 Install a client.

The following uses installing a Flink client on a node in the cluster as an example:

1. Log in to FusionInsight Manager, choose **Cluster**, click the name of the desired cluster, click **More**, and select **Download Client**.
2. In the **Download Cluster Client** dialog box that is displayed, select **Complete Client** for **Select Client Type**, select the platform type that matches the architecture of the node where the client is to install, select **Save to Path**, and click **OK**.
   - The generated file is stored in the **/tmp/FusionInsight-Client** directory on the active management node by default.
   - The name of the client software package is in the following format: **FusionInsight_Cluster_*<Cluster ID>*_Services_Client.tar**. In this section, the cluster ID **1** is used as an example. Replace it with the actual cluster ID.
3. Log in to the server where the client is to be installed as the client installation user.
4. Go to the directory where the installation package is stored and run the following commands to decompress the package:

   cd /tmp/FusionInsight-Client

   tar -xvf FusionInsight_Cluster_1_Services_Client.tar
5. Run the following command to verify the decompressed file and check whether the command output is consistent with the information in the **sha256** file:

   sha256sum -c FusionInsight_Cluster_1_Services_ClientConfig.tar.sha256

   ```
   FusionInsight_Cluster_1_Services_ClientConfig.tar: OK
   ```
6. Decompress the obtained installation file.

   tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar
7. Go to the directory where the installation package is stored, and run the following command to install the client to a specified directory (absolute path), for example, **/opt/hadoopclient**:

   cd /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Services_ClientConfig

   ./install.sh /opt/hadoopclient

   The client is installed if information similar to the following is displayed:

   ```
   The component client is installed successfully
   ```

Step 2 Log in to the node where the client is installed as the client installation user.

Step 3 Run the following command to go to the client installation directory:

cd /opt/hadoopclient

Step 4 Run the following command to initialize environment variables:

source /opt/hadoopclient/bigdata_env

Step 5 Perform the following operations if Kerberos authentication is enabled for the cluster. Otherwise, skip these operations.

1. Prepare a user for submitting Flink jobs.

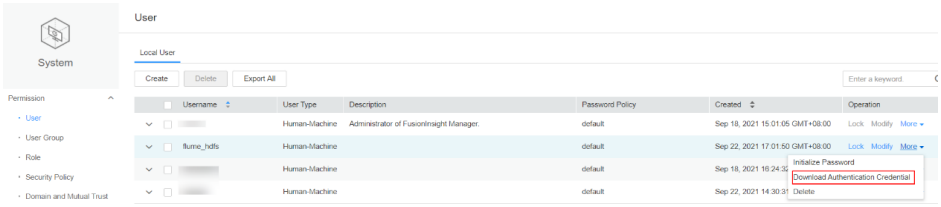
   Log in to FusionInsight Manager and choose **System** > **Permission** > **Role**. Click **Create Role** and configure **Role Name** and **Description**. In **Configure Resource Permission**, choose *Name of the desired cluster* > **Flink** and select **FlinkServer Admin Privilege**. Then click **OK**.

   Choose **System** > **Permission** > **User** and click **Create User**. Configure **Username**, set **User Type** to **Human-Machine**, configure **Password** and **Confirm Password**, click **Add** next to **User Group** to add the **hadoop**, **yarnviewgroup**, and **hadooppmanager** user groups as needed, click **Add** next to **Role** to add the **System_administrator**, **default**, and the created role, and click **OK**. (If you create a Flink job user for the first time, log in to FusionInsight Manager as the user and change the password.)

2. Log in to Manager and download the authentication credential.

   Log in to Manager and choose **System** > **Permission** > **User**. On the displayed page, locate the row that contains the added user, click **More** in the **Operation** column, and select **Download Authentication Credential**.

   Figure 6-4 Downloading the authentication credential

3. Decompress the downloaded authentication credential package and copy the obtained file to the client node, for example, the **/opt/hadoopclient/Flink/flink/conf** directory. If the client is installed on a node outside the cluster, copy the obtained files to the **/etc/** directory on the node.

4. Add the service IP address of the node where the client is installed and IP address of the master node to the **jobmanager.web.access-control-allow-origin** and **jobmanager.web.allow-access-address** configuration items in the **/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml** file. Use commas (,) to separate the IP addresses.

   jobmanager.web.access-control-allow-origin: *xx.xx.xxx.xxx,xx.xx.xxx.xxx,xx.xx.xxx.xxx*
   jobmanager.web.allow-access-address: *xx.xx.xxx.xxx,xx.xx.xxx.xxx,xx.xx.xxx.xxx*

NOTE

To obtain the service IP address of the node where the client is installed, perform the following operations:

- Node inside the cluster:

  In the navigation pane of the MRS management console, choose **Active Clusters**, select a cluster, and click its name to switch to the cluster details page.

  On the **Nodes** tab page, view the IP address of the node where the client is installed.

- Node outside the cluster: IP address of the ECS where the client is installed.

5. Configure security authentication by adding the **keytab** path and username in the **/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml** configuration file.

   security.kerberos.login.keytab: *<user.keytab file path>*
   security.kerberos.login.principal: *<Username>*

   Example:

   ```
   security.kerberos.login.keytab: /opt/hadoopclient/Flink/flink/conf/user.keytab
   security.kerberos.login.principal: test
   ```

6. In the **bin** directory of the Flink client, run the following commands to perform security hardening. Then, set a password for submitting jobs.

   cd /opt/hadoopclient/Flink/flink/bin

   sh generate_keystore.sh

   The script automatically replaces the SSL value in the **/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml** file.

   NOTE

   After authentication and encryption, the **flink.keystore** and **flink.truststore** files are generated in the **conf** directory on the Flink client and the following configuration items are set to the default values in the **flink-conf.yaml** file:

   - Set **security.ssl.keystore** to the absolute path of the **flink.keystore** file.
   - Set **security.ssl.truststore** to the absolute path of the **flink.truststore** file.
   - Set **security.cookie** to a random password automatically generated by the **generate_keystore.sh** script.
   - By default, **security.ssl.encrypt.enabled** is set to **false** in the **flink-conf.yaml** file by default. The **generate_keystore.sh** script sets **security.ssl.key-password**, **security.ssl.keystore-password**, and **security.ssl.truststore-password** to the password entered when the **generate_keystore.sh** script is called.
   - If ciphertext is required and **security.ssl.encrypt.enabled** is **true** in the **flink-conf.yaml** file, the **generate_keystore.sh** script does not set **security.ssl.key-password**, **security.ssl.keystore-password**, and **security.ssl.truststore-password**. To obtain the values, use the Manager plaintext encryption API by running the following command: **curl -k -i -u** *Username*:*Password* **-X POST -HContent-type:application/json -d '{"plainText":"***Password***"}' 'https://***x.x.x.x***:28443/web/api/v2/tools/encrypt'**

     In the preceding command, *Username*:*Password* indicates the user name and password for logging in to the system. The password of **"plainText"** indicates the one used to call the **generate_keystore.sh** script. *x.x.x.x* indicates the floating IP address of Manager. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

7. Configure paths for the client to access the **flink.keystore** and **flink.truststore** files.

   - Relative path (recommended):

Perform the following steps to set the file path of **flink.keystore** and **flink.truststore** to the relative path and ensure that the directory where the Flink client command is executed can directly access the relative paths.

i. Create a directory, for example, **ssl**, in **/opt/hadoopclient/Flink/flink/conf/**.

cd /opt/hadoopclient/Flink/flink/conf/

mkdir ssl

ii. Move the **flink.keystore** and **flink.truststore** files to the **/opt/hadoopclient/Flink/flink/conf/ssl/** directory.

mv flink.keystore ssl/

mv flink.truststore ssl/

iii. Change the values of the following parameters to relative paths in the **flink-conf.yaml** file:

```
security.ssl.keystore: ssl/flink.keystore
security.ssl.truststore: ssl/flink.truststore
```

– Absolute path:

After the **generate_keystore.sh** script is executed, the file path of **flink.keystore** and **flink.truststore** is automatically set to the absolute path **/opt/hadoopclient/Flink/flink/conf/** in the **flink-conf.yaml** file. In this case, you need to move the **flink.keystore** and **flink.truststore** files from the **conf** directory to this absolute path on the Flink client and YARN nodes.

Step 6 Run a wordcount job.

NOTICE

To submit or run jobs on Flink, the user must have the following permissions:

- If Ranger authentication is enabled, the current user must belong to the **hadoop** group or the user has been granted the **/flink** read and write permissions in Ranger.
- If Ranger authentication is disabled, the current user must belong to the **hadoop** group.

- For a normal cluster (Kerberos authentication disabled), you can submit jobs in either of the following ways:
  – Run the following commands to start a session and submit a job in the session:

    yarn-session.sh -nm "*session-name***" -d**

    flink run /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar

  – Run the following command to submit a single job on Yarn:

    flink run -m yarn-cluster /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
- For a security cluster (Kerberos authentication enabled), you can submit jobs in either of the following ways based on the paths of the **flink.keystore** and **flink.truststore** files:

- If the **flink.keystore** and **flink.truststore** files are stored in the relative path:
  - Run the following command in the directory at the same level as **ssl** to start the session and submit the job in the session:

    ssl is a relative path. For example, if **ssl** is in **opt/hadoopclient/Flink/flink/conf/**, run the command in the **opt/hadoopclient/Flink/flink/conf/** directory.

    cd /opt/hadoopclient/Flink/flink/conf

    yarn-session.sh -t ssl/ -nm "*session-name***" -d**

    flink run /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
  - Run the following command to submit a single job on Yarn:

    cd /opt/hadoopclient/Flink/flink/conf

    flink run -m yarn-cluster -yt ssl/ /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
- If the **flink.keystore** and **flink.truststore** files are stored in the absolute path:
  - Run the following commands to start a session and submit a job in the session:

    cd /opt/hadoopclient/Flink/flink/conf

    yarn-session.sh -nm "*session-name***" -d**

    flink run /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
  - Run the following command to submit a single job on Yarn:

    flink run -m yarn-cluster /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar

Step 7 After the job has been successfully submitted, the following information is displayed on the client:

Figure 6-5 Job submitted successfully on Yarn

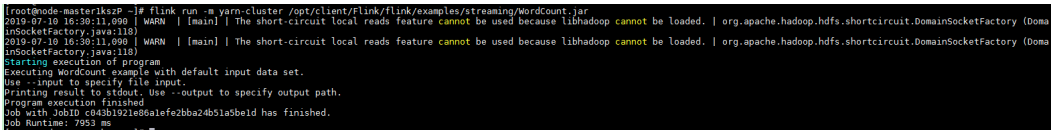


Figure 6-6 Session started successfully

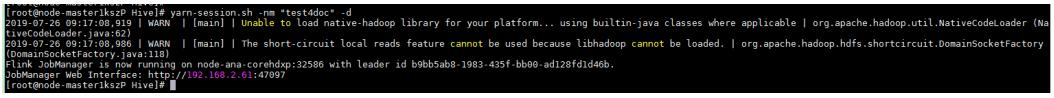
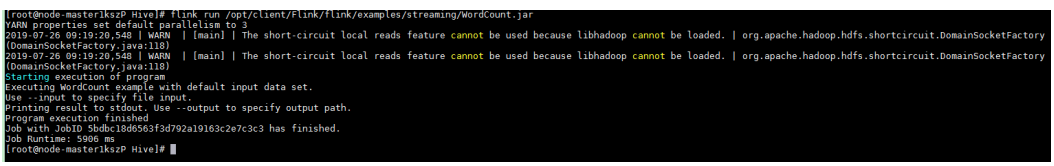


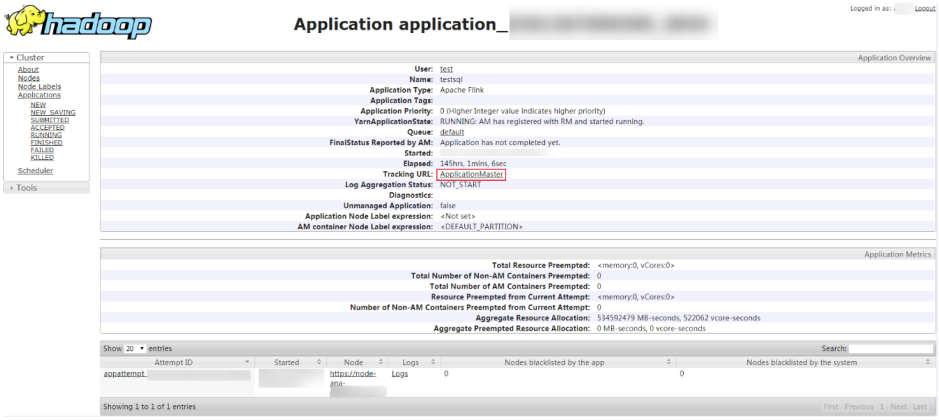
Figure 6-7 Job submitted successfully in the session



Step 8 Go to the native page of the YARN service, find the application of the corresponding job, and click the application name to go to the job details page.

- If the job is not completed, click **Tracking URL** to go to the native Flink page and view the job running information.
- If the job submitted in a session has been completed, you can click **Tracking URL** to log in to the native Flink service page to view job information.

Figure 6-8 application



----End

6.4 Preparing for Creating a FlinkServer Job

6.4.1 Accessing the FlinkServer Web UI

Scenario

After Flink is installed in an MRS cluster, you can connect to clusters and data as well as manage stream tables and jobs using the Flink web UI.

This section describes how to access the Flink web UI in an MRS cluster. Site trust must be added to the browser when you access Manager and the Flink web UI for the first time. Otherwise, the Flink web UI cannot be accessed.

Accessing the FlinkServer Web UI

Step 1 Log in to FusionInsight Manager as a user with **FlinkServer Admin Privilege**. Choose **Cluster** > **Services** > **Flink**.

NOTE

If your MRS cluster requires Kerberos authentication, create a role with the FlinkServer administrator permission or the application viewing and editing permission, and associate the role to the user. Then, you can access the Flink web UI. For details about how to create a role, see **Creating a FlinkServer Role**.

Step 2 On the right of **Flink WebUI**, click the link to access the Flink web UI.

Figure 6-9 Accessing the Flink Web UI

The Flink web UI provides the following functions:

- System management:
  - Cluster connection management allows you to create, view, edit, test, and delete a cluster connection.
  - Data connection management allows you to create, view, edit, test, and delete a data connection. Data connection types include HDFS and Kafka.
  - Application management allows you to create, view, and delete an application.
- Stream table management allows you to create, view, edit, and delete a stream table.
- Job management allows you to create, view, start, develop, edit, stop, and delete a job.

----End

6.4.2 Creating a FlinkServer Application

Applications can be used to isolate different upper-layer services.

Step 1 Access the Flink web UI as a user with **FlinkServer Admin Privilege**. For details, see **Accessing the FlinkServer Web UI**.

Step 2 Choose **System Management** > **Application Management**.

Step 3 Click **Create Application**. On the displayed page, set parameters and click **OK**.

After the application is created, you can switch to the application to be operated in the upper left corner of the Flink web UI and develop jobs.

----End

6.4.3 Creating a FlinkServer Cluster Connection

Different clusters can be accessed by configuring the cluster connection.

Procedure

Step 1 Access the Flink web UI. For details, see **Accessing the FlinkServer Web UI**.

Step 2 Choose **System Management** > **Cluster Connection Management**. The **Cluster Connection Management** page is displayed.

Step 3 Click **Create Cluster Connection**. On the displayed page, set parameters by referring to **Table 6-9** and click **OK**. After the cluster connection is created, you can edit, test, or delete the cluster connection in the **Operation** column.

Figure 6-10 Creating a cluster connection

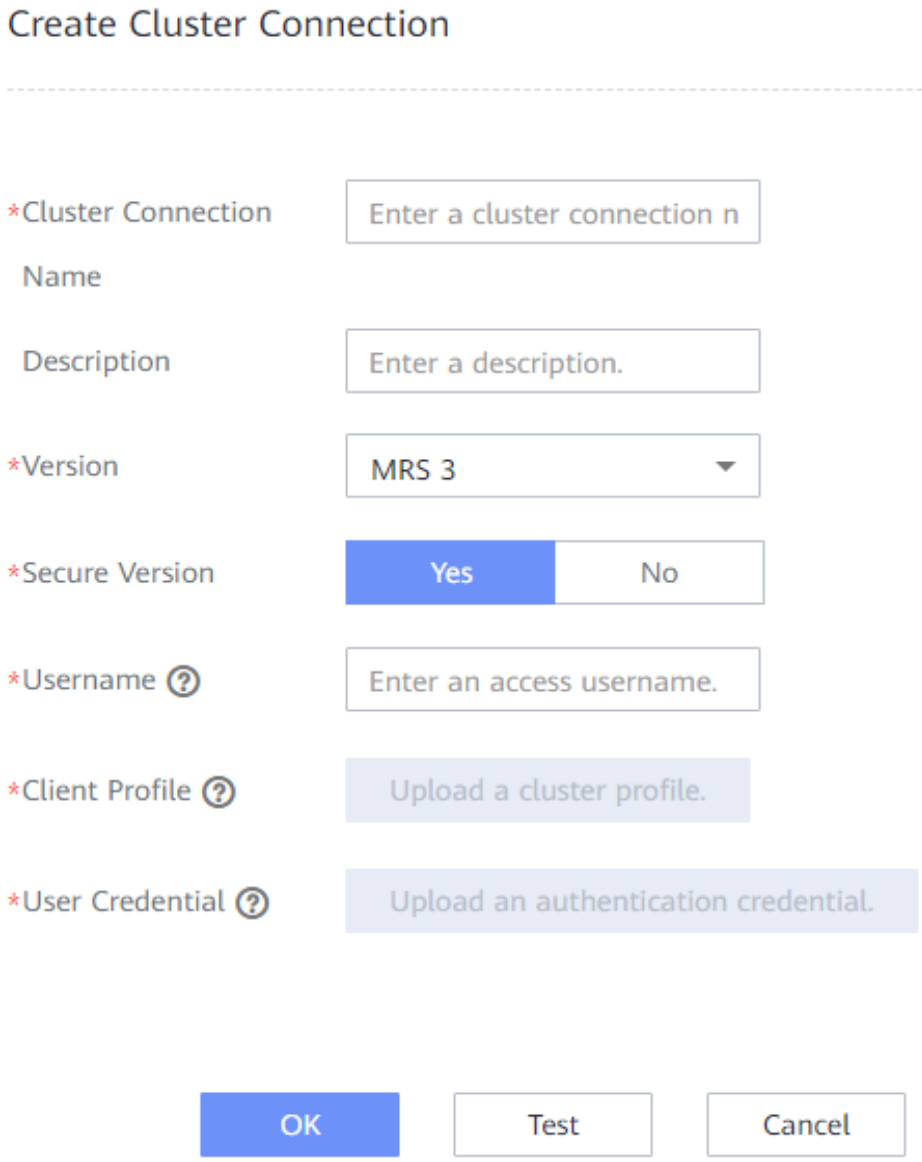


Table 6-9 Parameters for creating a cluster connection

Parameter	Description
Cluster Connection Name	Enter the name of the cluster connection.
Description	Enter the description of the cluster connection name.
FusionInsight HD Version	Set a cluster version.
Secure Version	• If the secure version is used, select **Yes** for a security cluster. Enter the username and upload the user credential. • If not, select **No**.
Username	The user must have the minimum permissions for accessing services in the cluster. This parameter is available only when **Secure Version** is set to **Yes**.
Client Profile	Client profile of the cluster, in TAR format.
User Credential	User authentication credential in FusionInsight Manager in TAR format. This parameter is available only when **Secure Version** is set to **Yes**. Files can be uploaded only after the username is entered.

NOTE

To obtain the cluster client configuration files, perform the following steps:

1. Log in to FusionInsight Manager and choose **Cluster** > **Dashboard**.
2. Choose **More** > **Download Client** > **Configuration Files Only**, select a platform type, and click **OK**.

To obtain the user credential, perform the following steps:

1. Log in to FusionInsight Manager and click **System**.
2. In the **Operation** column of the user, choose **More** > **Download Authentication Credential**, select a cluster, and click **OK**.

----End

6.4.4 Creating a FlinkServer Data Connection

You can use data connections to access different data services. Currently, FlinkServer supports HDFS, Redis, and Kafka data connections.

Procedure

Step 1 Access the Flink web UI. For details, see **Accessing the FlinkServer Web UI**.

Step 2 Choose **System Management** > **Data Connection Management**. The **Data Connection Management** page is displayed.

Step 3 Click **Create Data Connection**. On the displayed page, select a data connection type, enter information by referring to **Table 6-10**, and click **OK**. After the data connection is created, you can edit, test, or delete the data connection in the **Operation** column.

Table 6-10 Parameters for creating a data connection

Parameter	Description	Example Value
Data Connection Type	Type of the data connection, which can be **HDFS**, **Redis**, or **Kafka**. If you select the Redis data connection type, prepare a DCS (for Redis) instance in advance. Ensure that **Instance Type** is **Redis Cluster**, **Password Protected** is **No**, and **Region** and **VPC** are the same as those of the cluster where Flink is located.	-
Data Connection Name	Name of the data connection.	-
Cluster Connection	Cluster connection name in configuration management. This parameter is mandatory for HDFS data connections.	-
Kafka broker	Connection information about Kafka broker instances. The format is *IP address*:*Port number*. Use commas (,) to separate multiple instances. This parameter is mandatory for Kafka data connections.	192.168.0.1:21005,192.168.0.2:21005
Redis Deployment Method	Redis deployment mode. Currently, only **Cluster** is supported. This parameter is mandatory for Redis data connections.	Cluster
Redis Server List	Connection information about Redis instances. The format is *IP address*:*Port number*. Use commas (,) to separate multiple instances. This parameter is mandatory for Redis data connections.	192.168.0.1:6379,192.168.0.2:6379

Parameter	Description	Example Value
Authentication Mode	• **SIMPLE**: indicates that the connected service is in non-security mode and does not need to be authenticated. The Kafka data connection created using this method supports only the SIMPLE authentication type. • **KERBEROS**: indicates that the connected service is in security mode and the Kerberos protocol for security authentication is used for authentication.	-

----End

6.4.5 Creating a FlinkServer Stream Table Source

Data tables can be used to define basic attributes and parameters of source tables, dimension tables, and output tables.

Procedure

Step 1 Access the Flink web UI. For details, see **Accessing the FlinkServer Web UI**.

Step 2 Click **Table Management**. The table management page is displayed.

Step 3 Click **Create Stream Table**. On the stream table creation page, set parameters by referring to **Table 6-11** and click **OK**. After the stream table is created, you can edit or delete the stream table in the **Operation** column.

Figure 6-11 Creating a stream table

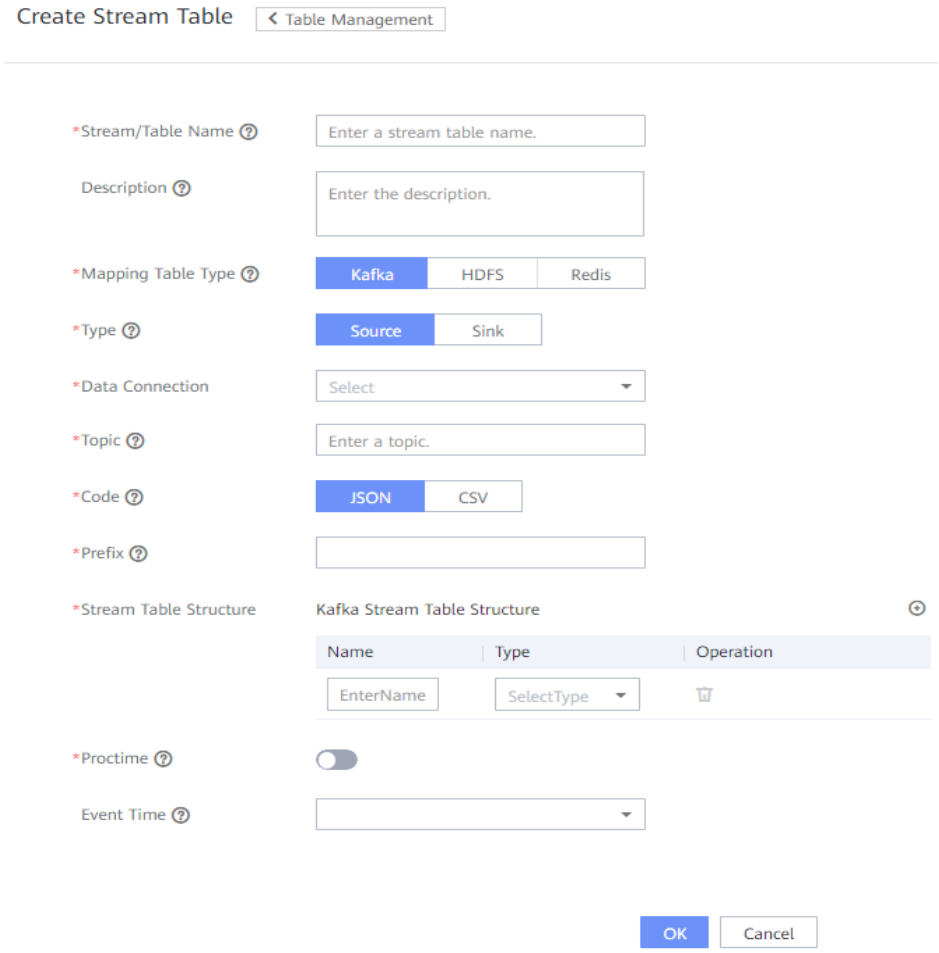


Table 6-11 Parameters for creating a stream table

Parameter	Description	Remarks
Stream/Table Name	Stream/Table name	Example: **flink_sink**
Description	Stream/Table description information	-
Mapping Table Type	Flink SQL does not provide the data storage function. Table creation is actually the creation of mapping for external data tables or storage. The value can be **Kafka** or **HDFS**.	-
Type	Includes data source table **Source** and data result table **Sink**. Tables included in different mapping table types are as follows: • Kafka: **Source** and **Sink** • HDFS: **Source** and **Sink**	-

Parameter	Description	Remarks
Data Connection	Name of the data connection	-
Topic	Kafka topic to be read. Multiple Kafka topics can be read. Use separators to separate topics. This parameter is available when **Mapping Table Type** is set to **Kafka**.	-
File Path	HDFS directory or a single file path to be transferred. This parameter is available when **Mapping Table Type** is set to **HDFS**.	Example: **/user/sqoop/** or **/user/sqoop/example.csv**
Code	Codes corresponding to different mapping table types are as follows: ● Kafka: **CSV** and **JSON** ● HDFS: **CSV**	-
Prefix	When **Mapping Table Type** is set to **Kafka**, **Type** is set to **Source**, and **Code** is set to **JSON**, this parameter indicates the hierarchical prefixes of multi-layer nested JSON, which are separated by commas (,).	For example, **data,info** indicates that the content under **data** and **info** in the nested JSON file is used as the data input in JSON format.
Separator	Has different meanings when **Mapping Table Type** is set to the following values: It is used as the separator of specified CSV fields. This parameter is available only when **Code** is set to **CSV**.	Example: comma (**,**)
Row Separator	Line break in the file, including **\r**, **\n**, and **\r\n**. This parameter is available when **Mapping Table Type** is set to **HDFS**.	-
Column Separator	Field separator in the file. This parameter is available when **Mapping Table Type** is set to **HDFS**.	Example: comma (**,**)
Stream Table Structure	Stream/Table structure, including **Name** and **Type**.	-

Parameter	Description	Remarks
Proctime	System time, which is irrelevant to the data timestamp. That is, the time when the calculation is complete in Flink operators. This parameter is available when **Type** is set to **Source**.	-
Event Time	Time when an event is generated, that is, the timestamp generated during data generation. This parameter is available when **Type** is set to **Source**.	-

----End

6.5 Creating a FlinkServer Job

6.5.1 Creating a FlinkServer Job and Writing Data to a ClickHouse Table

This section applies to MRS 3.1.2 or later clusters.

Scenario

Flink interconnects with the ClickHouseBalancer instance of ClickHouse to read and write data, preventing ClickHouse traffic distribution problems. The following table lists the mapping between Flink SQL and ClickHouse data types.

NOTICE

When "FlinkSQL" is displayed in the command output on the FlinkServer web UI in MRS 3.2.0 or later clusters, the **password** field in the SQL statement is left blank to meet security requirements. Before you submit a job, manually enter the password.

Table 6-12 Mapping between Flink SQL and ClickHouse data types

Flink SQL Data Type	ClickHouse Data Type
BOOLEAN	UInt8
TINYINT	Int8
SMALLINT	Int16
INTEGER	Int32

Flink SQL Data Type	ClickHouse Data Type
BIGINT	Int64
FLOAT	Float32
DOUBLE	Float64
CHAR	String
VARCHAR	String
VARBINARY	FixedString
DATE	Date
TIMESTAMP	DateTime
DECIMAL	Decimal

Prerequisites

- Services such as ClickHouse, HDFS, YARN, Flink, and Kafka have been installed in the cluster.
- The client has been installed, for example, in **/opt/client**.

Creating a Job

Step 1 Log in to the node where the client is installed as user **root**.

Step 2 Run the following command to go to the client installation directory:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the current user. The current user must have the permission to create ClickHouse tables. If Kerberos authentication is disabled for the current cluster, skip this step:

kinit *Component service user*

Example: **kinit clickhouseuser**

Step 5 Connect to the ClickHouse client. For details, see **ClickHouse Client Practices**. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

- Normal mode:

  clickhouse client --host *IP address of the ClickHouse instance* **--user** *Username* **--password** '*Password*' **--port** *ClickHouse port number* **--multiline**

- Security mode:

clickhouse client --host *IP address of the ClickHouse instance* **--user** *Username* **--password** '*Password*'**--port** *ClickHouse port number* **--secure --multiline**

Step 6 Run the following statements to create a replication table and a distributed table.

1. Create a replication table **default.test1**.

```
CREATE TABLE default.test1 on cluster default_cluster
(
`pid` Int8,
`uid` UInt8,
`Int_16` Int16,
`Int_32` Int32,
`Int_64` Int64,
`String_x` String,
`String_y` String,
`float_32` Float32,
`float_64` Float64,
`Decimal_x` Decimal32(2),
`Date_x` Date,
`DateTime_x` DateTime
)
ENGINE = ReplicatedReplacingMergeTree('/clickhouse/tables/{shard}/test1','{replica}')
PARTITION BY pid
ORDER BY (pid, DateTime_x);
```

2. Create a distributed table **test1_all**.

```
CREATE TABLE test1_all ON CLUSTER default_cluster
(
`pid` Int8,
`uid` UInt8,
`Int_16` Int16,
`Int_32` Int32,
`Int_64` Int64,
`String_x` String,
`String_y` String,
`float_32` Float32,
`float_64` Float64,
`Decimal_x` Decimal32(2),
`Date_x` Date,
`DateTime_x` DateTime
)
ENGINE = Distributed(default_cluster, default, test1, rand());
```

Step 7 Log in to Manager and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link on the right of **Flink WebUI** to access the Flink web UI.

Step 8 Create a Flink SQL job and set **Task Type** to **Stream job**. For details, see **Creating a FlinkServer Job**. On the job development page, configure the job parameters as follows and start the job. In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **60000**, and retain the default value for **Mode**.

- If the current MRS cluster is in security mode, perform the following operations:

```
create table kafkasource(
`pid` TINYINT,
`uid` BOOLEAN,
`Int_16` SMALLINT,
`Int_32` INTEGER,
`Int_64` BIGINT,
`String_x` CHAR,
`String_y` VARCHAR(10),
`float_32` FLOAT,
`float_64` DOUBLE,
`Decimal_x` DECIMAL(9,2),
`Date_x` DATE,
```



```
`DateTime_x` TIMESTAMP
) with(
  'connector' = 'kafka',
  'topic' = 'input',
  'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port number',
  'properties.group.id' = 'group1',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.System domain name'
);
CREATE TABLE cksink (
`pid` TINYINT,
`uid` BOOLEAN,
`Int_16` SMALLINT,
`Int_32` INTEGER,
`Int_64` BIGINT,
`String_x` CHAR,
`String_y` VARCHAR(10),
`float_32` FLOAT,
`float_64` DOUBLE,
`Decimal_x` DECIMAL(9,2),
`Date_x` DATE,
`DateTime_x` TIMESTAMP
) WITH (
'connector' = 'jdbc',
'url' = 'jdbc:clickhouse://IP address 1 of the ClickHouseBalancer instance:ClickHouseBalancer port number,IP address 2 of the ClickHouseBalancer instance:ClickHouseBalancer port number/default?
ssl=true&sslmode=none',
'username' = 'ClickHouse user. For details, see the note below.',
'password' = 'ClickHouse user password. For details, see the note below.',
'table-name' = 'test1_all',
'driver' = 'com.clickhouse.jdbc.ClickHouseDriver',
'sink.buffer-flush.max-rows' = '0',
'sink.buffer-flush.interval' = '60s'
);
Insert into cksink
select
*
from
kafkasource;
```

- If the current MRS cluster is in normal mode, perform the following operations:

```
create table kafkasource(
`pid` TINYINT,
`uid` BOOLEAN,
`Int_16` SMALLINT,
`Int_32` INTEGER,
`Int_64` BIGINT,
`String_x` CHAR,
`String_y` VARCHAR(10),
`float_32` FLOAT,
`float_64` DOUBLE,
`Decimal_x` DECIMAL(9,2),
`Date_x` DATE,
`DateTime_x` TIMESTAMP
) with(
  'connector' = 'kafka',
  'topic' = 'kinput',
  'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port number',
  'properties.group.id' = 'kafka_test',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);
CREATE TABLE cksink (
`pid` TINYINT,
`uid` BOOLEAN,
`Int_16` SMALLINT,
```



```
`Int_32` INTEGER,
`Int_64` BIGINT,
`String_x` CHAR,
`String_y` VARCHAR(10),
`float_32` FLOAT,
`float_64` DOUBLE,
`Decimal_x` DECIMAL(9,2),
`Date_x` DATE,
`DateTime_x` TIMESTAMP
) WITH (
'connector' = 'jdbc',
'url' = 'jdbc:clickhouse://IP address 1 of the ClickHouseBalancer instance:ClickHouseBalancer port number,IP address 2 of the ClickHouseBalancer instance:ClickHouseBalancer port number/default',
'table-name' = 'test1_all',
'username' = 'ClickHouse user. For details, see the note below.',
'password' = 'ClickHouse user password. For details, see the note below.',
'driver' = 'com.clickhouse.jdbc.ClickHouseDriver',
'sink.buffer-flush.max-rows' = '0',
'sink.buffer-flush.interval' = '60s'
);
Insert into cksink
select
*
from
kafkasource;
```


NOTE

- If an MRS cluster is in the security mode, the user in the **cksink** table must have related permissions on the ClickHouse tables. For details, see **Creating a ClickHouse Role**.
- Kafka port number
  - Value of **sasl.port** when **Authentication Mode** of the cluster is **Security Mode**, **21007** by default.
  - Value of **port** when **Authentication Mode** of the cluster is **Normal Mode**, **9092** by default.

    If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. On the page that is displayed, click the **Configurations** tab then the **All Configurations** sub-tab. On the displayed page, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- *System domain name*: You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.
- ClickHouseBalancer port:
  - If the cluster where ClickHouse is deployed is in security mode, the ClickHouseBalancer port is **21426** by default.
  - If the cluster where ClickHouse is deployed is in normal mode, the ClickHouseBalancer port is **21425** by default.
- **url**: You can configure multiple IP addresses for ClickHouseBalancer instances to avoid single points of failure (SPOFs) of the instances.
- DELETE messages generated during Flink computing are filtered out when data is written to ClickHouse.
- Parameters for batch write: Flink stores data in the memory and then flushes the data to the database table when the trigger condition is met. The configurations are as follows:

  sink.buffer-flush.max-rows: Number of rows written to ClickHouse. The default value is **100**.

  sink.buffer-flush.interval: Interval for batch write. The default value is **1s**.

  If either of the two conditions is met, a sink operation is triggered. That is, data will be flushed to the database table.
  - Scenario 1: sink every 60s

    'sink.buffer-flush.max-rows' = '0',

    'sink.buffer-flush.interval' = '60s'
  - Scenario 2: sink every 100 rows

    'sink.buffer-flush.max-rows' = '100',

    'sink.buffer-flush.interval' = '0s'
  - Scenario 3: no sink

    'sink.buffer-flush.max-rows' = '0',

    'sink.buffer-flush.interval' = '0s'

Step 9 On the job management page, check whether the job status is **Running**.

Step 10 Execute the following script to write data to Kafka. For details, see **Managing Messages in Kafka Topics**.

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka instances reside*:*Kafka port number* **--topic***Topic name* **--producer.config** *Client directory*/**Kafka/kafka/config/producer.properties**

For example, if the topic name is **kinput**, the script is **sh kafka-console-producer.sh --broker-list** *IP address of the node where the Kafka instance is deployed*:*Kafka port* **--topic kinput --producer.config /opt/client/Kafka/kafka/config/producer.properties**

Enter the message content.

```
{"pid": "3","uid":false,"Int_16": "6533","Int_32": "429496294","Int_64": "1844674407370955614","String_x": "abc1","String_y": "abc1defghi","float_32": "0.1234","float_64": "95.1","Decimal_x": "0.451236414","Date_x": "2021-05-29","DateTime_x": "2021-05-21 10:05:10"}
{"pid": "4","uid":false,"Int_16": "6533","Int_32": "429496294","Int_64": "1844674407370955614","String_x": "abc1","String_y": "abc1defghi","float_32": "0.1234","float_64": "95.1","Decimal_x": "0.4512314","Date_x": "2021-05-29","DateTime_x": "2021-05-21 10:05:10"}
```

Press **Enter** to send the message.

Step 11 Interconnect with ClickHouse to query the table data.

clickhouse client --host *IP address of the ClickHouse instance* **--user** *Username* **--password '***Password***' --port** *ClickHouse port number* **--secure --multiline**

Run the following command to check whether data is written to a specified ClickHouse table, for example, **test1_all**.

select * from test1_all;

----End

6.5.2 Creating a FlinkServer Job to Interconnect with a Doris Table

This section applies to MRS 3.5.0 and later.

Scenario

This topic describes how to use FlinkServer to write Kafka data to Doris and how to perform Lookup Join on Doris and Kafka data.

Prerequisites

- Services such as Doris, HDFS, YARN, Flink, and Kafka have been installed in the cluster.
- The node to be connected to the Doris database can communicate with the MRS cluster.
- A user with the Doris management permission has been created.
  - Kerberos authentication is enabled for the cluster (the cluster is in security mode)

    On FusionInsight Manager, create a human-machine user, for example, **dorisuser**, create a role with Doris administrator permission, and bind the role to the user.

    Log in to FusionInsight Manager as the new user **dorisuser** and change the initial password.
  - Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

    After connecting to Doris as user **admin**, create a role with administrator permissions, and bind the role to the user.

- The MySQL client has been installed. For details, see **Using the MySQL Client to Connect to Doris**.
- The Flink client has been installed.

Using FlinkServer to Write Kafka Data to Doris

Operations on the Doris side

Step 1 Log in to the node where MySQL is installed and run the following command to connect to the Doris database:

If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to connect to the Doris database:

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*Database login user* **-p -P***Connection port for FE queries* **-h***IP address of the Doris FE instance*

Enter the password for logging in to the database.

NOTE

- To obtain the query connection port of the Doris FE instance, you can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- To obtain the query connection port of the Doris FE instance, you can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **balancer_tcp_port** of the Doris service.
- To obtain the IP address of the Doris FE or DBalancer instance, log in to FusionInsight Manager of the MRS cluster and choose **Cluster** > **Services** > **Doris** > **Instances** to view the service IP address of any FE or DBalancer instance.
- You can also use the MySQL connection software or Doris web UI to connect to the database.

Step 2 Run the following commands to create a database and table **usertable2** to which data is written:

create database *sink*;

use *sink*;

create table *usertable2*(

`user_id` VARCHAR(10),

`user_name` VARCHAR(10),

`age` INT

)

DISTRIBUTED BY HASH(user_id) BUCKETS 32;

Operations on the Flink side

Step 3 Log in to FusionInsight Manager as a user with **FlinkServer Admin Privilege**. Choose **Cluster** > **Services** > **Flink**.

NOTE

If your MRS cluster requires Kerberos authentication, create a role with the FlinkServer administrator permission or the application viewing and editing permission, and bind the role to the user. Then, you can access the Flink Web UI. For details about how to create a role, see **Creating a FlinkServer Role**.

Step 4 On the right of Flink web UI, click the link to access FlinkServer.

Step 5 On the FlinkServer page, choose **Job Management** > **Create Job**. On the displayed dialog box, set the following parameters and click **OK**. The **Job Management** page is displayed.

- **Type**: Select **Flink SQL**.
- **Name**: Enter a job name, for example, **FlinkSQL1**.

Step 6 Create a stream or batch Flink SQL job on the Flink job management page. The following are some examples:

1. Create a Kafka data source table.

   CREATE TABLE *KafkaSource* **(**
   `user_id` VARCHAR,
   `user_name` VARCHAR,
   `age` INT
) WITH (
   'connector' = 'kafka',
   'topic' ='*Topic name***',**
   'properties.bootstrap.servers' ='*Service IP address of the Kafka Broker instance:Port of the Broker instance***',**
   'properties.group.id' = 'testGroup',
   'scan.startup.mode' = 'latest-offset',
   'format' = 'csv',
   'properties.sasl.kerberos.service.name' = '*kafka***',**
   'properties.security.protocol' = '*SASL_PLAINTEXT***',**
   'properties.kerberos.domain.name' = '*hadoop.hadoop.com***'**
   );

   NOTE

   - **properties.bootstrap.servers**: If there are multiple parameter values, separate them with commas (,).

     To view the service IP address of the Kafka Broker instance, log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka** > **Instances**. To view the port of the Broker instance, click **Configurations**. If Kerberos authentication is enabled for the cluster (the cluster is in security mode), search for **sasl.port**. If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), search for **port**.
   - The values of **properties.sasl.kerberos.service.name**, **properties.security.protocol**, and **properties.kerberos.domain.name** can be obtained from *Client installation directory*/**Kafka/kafka/config** on the node where the Kafka client is installed. Search for **sasl.kerberos.service.name**, **security.protocol**, or **kerberos.domain.name** in the **server.properties** file in the directory.

2. Create a Doris Sink table:

   CREATE TABLE *dorisSink* **(**
   `user_id` VARCHAR,
   `user_name` VARCHAR,
   `age` INT
) WITH (
   'connector' = 'doris',
   'fenodes' ='*IP address of any FE instance:HTTPS port or HTTP port***',**
   'table.identifier' = '*sink.usertable2***',**
   'username' = '*user***',**
   'password' = '*password***',**
   'doris.enable.https' = 'true',
   'doris.ignore.https.ca' = 'true',
   'sink.label-prefix' = 'doris_label1'
   );

   NOTE

   - To view the IP addresses of FE instances, choose **Cluster** > **Services** > **Doris** > **Instances** on FusionInsight Manager.
   - To view the HTTPS port, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and search for **https_port**. To view the HTTP port, search for **http_port**.
   - **table.identifier**: The parameter value is the Doris database and table created in **Step 2**.
   - **username** and **password** are the username and password for connecting to the Doris.
   - After HTTPS is disabled for a cluster in normal or security mode, remove the following configuration parameters from the **with** clause of the Doris Sink table creation statement:

     'doris.enable.https' = 'true'

     'doris.ignore.https.ca' = 'true'
   - When creating a Doris Sink table, you can also set the parameters listed in **Table 6-13**.

3. Run the following command to write Kafka data to Doris:

   insert into *dorisSink* **select * from** *KafkaSource***;**

Step 7 In the basic parameters area on the right of the **Job Management** page, select **Enable CheckPoint** and set **Time Interval(ms)** as required. The recommended interval ranges from 30000 to 60000.

Step 8 Click **Check Semantic** to verify the semantics of the statements. Then, click **Save** and **Submit**.

Operations on the Kafka side

Step 9 Log in to the node where the Kafka client is installed and perform the following operations to create a Kafka topic:

cd *Client installation directory*/**Kafka/kafka/bin**

1. Run the following command to create a Kafka topic. The topic name must be the same as that configured in **6.1**:

   sh kafka-topics.sh --create --topic *Topic name* **--partitions 1 --replication-factor 1 --bootstrap-server** *IP address of the host where the Controller of the Kafka Broker instance is deployed*:*Port of the Broker instance* **--command-config ../config/client.properties**

2. Run the following command to view the topic list:

   sh kafka-topics.sh --list --bootstrap-server Kafka*IP address of the host where the Controller of the Broker instance is deployed:Port of the Broker instance* **--command-config ../config/client.properties**

3. Run the following command to connect to the Kafka client:

   sh kafka-console-producer.sh --broker-list *IP address of the host where the Controller of the Kafka Broker instance is deployed:Port of the Broker instance* **--topic TopicTest --producer.config ../config/producer.properties**

NOTE

To obtain the IP address of the host where the Controller of the Broker instance is deployed, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, and view the value of **Controller Host** in the basic information area on the **Dashboard** page.

Step 10 Connect to Doris on the node where the MySQL client is installed and run the following command to check whether the data in the Doris table is the same as the data inserted in **9.3**:

select * from *sink.usertable2*;

----End

Lookup Join

Step 1 Log in to FusionInsight Manager as a user with **FlinkServer Admin Privilege**. Choose **Cluster** > **Services** > **Flink**.

NOTE

If your MRS cluster requires Kerberos authentication, create a role with the FlinkServer administrator permission or the application viewing and editing permission, and bind the role to the user. Then, you can access the Flink Web UI. For details about how to create a role, see **Creating a FlinkServer Role**.

Step 2 On the right of Flink web UI, click the link to access FlinkServer.

Step 3 On the Flink web UI, click **Job Management** and then **Create Job**. In the **Create Job** dialog box, set the following parameters and click **OK**:

- **Type**: Select **Flink SQL**.
- **Name**: Enter a job name, for example, **FlinkSQL2**.

Step 4 Create a stream or batch Flink SQL job on the Flink job management page. The following are some examples:

1. Create a Kafka Source table.

   CREATE TABLE *fact_table* **(**

   `id` BIGINT,

   `name` STRING,

```
`city` STRING,
`process_time` as proctime()
) WITH (
'connector' = 'kafka',
'topic' ='Topic name',
'properties.bootstrap.servers' ='IP address of the Kafka Broker instance:21007',
'properties.group.id' = 'testGroup',
'scan.startup.mode' = 'latest-offset',
'format' = 'csv',
'properties.sasl.kerberos.service.name' = 'kafka',
'properties.security.protocol' = 'SASL_PLAINTEXT',
'properties.kerberos.domain.name' = 'hadoop.hadoop.com'
);
```

NOTE

- **properties.bootstrap.servers**: If there are multiple parameter values, separate them with commas (,).

  To view the service IP address of the Kafka Broker instance, log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka** > **Instances**. To view the port of the Broker instance, click **Configurations**. If Kerberos authentication is enabled for the cluster (the cluster is in security mode), search for **sasl.port**. If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), search for **port**.

- The values of **properties.sasl.kerberos.service.name**, **properties.security.protocol**, and **properties.kerberos.domain.name** can be obtained from *Client installation directory*/**Kafka/kafka/config** on the node where the Kafka client is installed. Search for **sasl.kerberos.service.name**, **security.protocol**, or **kerberos.domain.name** in the **server.properties** file in the directory.

2. Create a Flink table.

```
create table dim_city(
`city` STRING,
`level` INT ,
`province` STRING,
`country` STRING
) WITH (
'connector' = 'doris',
'fenodes' ='IP address of an FE instance:HTTPS port or HTTP port',
'jdbc-url' = 'jdbc:mysql://IP address of an FE instance:FE query connection port',
'table.identifier' = 'dim.dim_city',
'username' = 'user',
'password' = 'password'
);
```


NOTE

- To view the IP addresses of FE instances, log in to FusionInsight Manager and choose **Cluster** > **Services** > **Doris** > **Instances**.
- To view the HTTPS port, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and search for **https_port**. To view the HTTP port, search for **http_port**.
- To obtain the query connection port of the Doris FE instance, you can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Doris** > **Configurations**, and query the value of **query_port** of the Doris service.
- When creating a Flink table, you can also set the parameters listed in **Table 6-14**.

3. Run the following command to perform a join:

   SELECT a.id, a.name, a.city, c.province, c.country,c.level

   FROM *fact_table* **a**

   LEFT JOIN dim_city FOR SYSTEM_TIME AS OF a.process_time AS c

   ON a.city = c.city

Step 5 In the basic parameters area on the right of the **Job Management** page, select **Enable CheckPoint** and set **Time Interval(ms)** as required. The recommended interval ranges from 30000 to 60000.

Step 6 Click **Check Semantic** to verify the semantics of the statements. Then, click **Save** and **Submit**.

----End

Table Creation Configurations

Table 6-13 Optional parameters for creating a Doris Sink table

Parameter	Default Value	Mandatory	Description
doris.request.retries	3	No	Number of retry times for sending requests to Doris.
doris.request.connect.timeout.ms	30000	No	Connection timeout interval for sending requests to Doris.
doris.request.read.timeout.ms	30000	No	Read timeout interval for sending requests to Doris.
sink.max-retries	3	No	Maximum number of retry times after a commit is failed. The default value is 3.
sink.enable.batch-mode	false	No	Whether to write data to Doris in batch mode. If this is enabled, the write time does not depend on checkpoints. You can use **sink.buffer-flush.max-rows**, **sink.buffer-flush.max-bytes**, or **sink.buffer-flush.interval** to control the write time.

Parameter	Default Value	Mandatory	Description
sink.buffer-flush.max-rows	50000	No	Maximum number of data rows that can be written in a batch.
sink.buffer-flush.max-bytes	10MB	No	Maximum number of bytes that can be written in a batch.
sink.buffer-flush.interval	10s	No	Interval for asynchronously refreshing the cache in batches

Table 6-14 Optional parameters for creating a Flink table in Lookup Joins

Parameter	Default Value	Mandatory	Description
lookup.cache.max-rows	-1	No	Maximum number of rows that can be cached in the lookup table. The default value is **-1**, indicating that the cache function is disabled.
lookup.cache.ttl	10s	No	Maximum duration for caching lookup data. The default value is **10s**.
lookup.max-retries	1	No	Number of retry times after the lookup query fails
lookup.jdbc.async	false	No	Whether to enable asynchronous lookup. The default value is **false**.
lookup.jdbc.read.batch.size	128	No	Maximum batch size for each query when asynchronous lookup is enabled
lookup.jdbc.read.batch.queue-size	256	No	Size of the intermediate buffer queue when asynchronous lookup is enabled
lookup.jdbc.read.thread-size	3	No	Number of Lookup JDBC threads in each task

6.5.3 Creating a FlinkServer Job to Interconnect with a GaussDB(DWS) Table

This section applies to MRS 3.2.0 and to MRS 3.3.1.

Scenario

FlinkServer can interconnect with GaussDB(DWS) 8.1.*x* or later. This section describes the DDL definitions when GaussDB(DWS) serves as source tables, sink

tables, and dimension tables, as well as the **WITH** parameter and code examples used during table creation, and describes how to perform operations on the FlinkServer job management page. The following table lists the mapping between Flink SQL and GaussDB(DWS) data types.

In this example, FlinkServer and Kafka in security mode are used to interconnect with GaussDB(DWS) in security mode.

NOTICE

When "FlinkSQL" is displayed in the command output on the FlinkServer web UI, the **password** field in the SQL statement is left blank to meet security requirements. Before you submit a job, manually enter the password.

Table 6-15 Mappings Between Flink SQL and GaussDB(DWS) Data Types

Flink SQL Data Type	GaussDB(DWS) Data Type
BOOLEAN	BOOLEAN
TINYINT	-
SMALLINT	SMALLINT(INT2)
	SMALLSERIAL(SERIAL2)
INTEGER	INTEGER
	SERIAL
BIGINT	BIGINT
	BIGSERIAL
FLOAT	REAL
	FLOAT4
DOUBLE	DOUBLE
	FLOAT8
CHAR	CHAR(n)
VARCHAR	VARCHAR(n)
DATE	DATE
TIMESTAMP	TIMESTAMP[(p)] [WITHOUT TIME ZONE]
DECIMAL	NUMERIC[(p[,s])]
	DECIMAL[(p[,s])]

Prerequisites

- The cluster where FlinkServer is deployed must be able to communicate with the cluster where GaussDB(DWS) is deployed. The AZ, VPC, and security group configurations of the two clusters must be the same.
- Cluster where FlinkServer resides (security mode):
  - HDFS, YARN, Kafka, ZooKeeper, and Flink have been installed in the cluster.
  - The client that contains the Kafka service has been installed, for example, in the **/opt/client** directory.
  - You have created a user with the **FlinkServer Admin Privilege** (for example, **flinkuser**) for accessing the Flink web UI by referring to **Creating a FlinkServer Role**.
- Cluster of the GaussDB(DWS) to be interconnected (security mode):

  Run the following commands to connect to the database and create a table for receiving data:

  gsql -d *postgres* **-h** *IP* **-U** *username* **–p** *port* **-W** *password* **–r**

  NOTE

  - **postgres** indicates the name of the database to be connected.
  - **IP**: indicates the IP address of the GaussDB(DWS) cluster. If a public network address is used for connection, set this parameter to the public network domain name. If a private network address is used for connection, set this parameter to the private network domain name. If an ELB is used for connection, set this parameter to the ELB address.
  - **username** and **password** indicate the username and password for connecting to the database. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.
  - **port**: indicates the port number of the Coordinator. Replace it with the actual port number. You can run the **gs_om -t status --detail** command to query the Coordinator data path and view the port number in the **postgresql.conf** file in the path.

  Create an empty table for receiving data, for example, **customer_t1**.

```
CREATE TABLE customer_t1
(
    c_customer_sk             INTEGER,
    c_customer_name           VARCHAR(32)
)
with (orientation = column,compression=middle)
distribute by hash (c_customer_name);
```

GaussDB as a Sink Table

Step 1 Log in to Manager as user **flinkuser** and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link next to **Flink WebUI** to access the Flink web UI.

Step 2 Create a Flink SQL job by referring to **Creating a FlinkServer Job**. On the job development page, configure the job parameters as follows and start the job.

In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **60000**, and retain the default value for **Mode**.


```
CREATE TABLE MyUserTable(
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH(
  'connector' = 'jdbc',
  'url'='jdbc:gaussdb://IP address of the GaussDB server:Database port number/postgres',
  'table-name' = 'customer_t1',--If table customer_t1 is created in schema base, the configuration rule is 'table-name' = 'base."customer_t1"'.
  'username' = 'username',--Username for logging in to the GaussDB(DWS) database
  'password'='password'--Password for connecting to the GaussDB(DWS) database. You need to specify the password when submitting the job.
  'write.mode' = 'upsert',--When data is written in upsert mode, you can set whether to ignore null values. (applicable to MRS 3.3.0 and later versions)
  'ignoreNullWhenUpsert' = 'false'--true indicates that null values are ignored, and false indicates that null values are not ignored and written to the database.
);
CREATE TABLE KafkaSource (
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH (
  'connector' = 'kafka',
  'topic' = 'customer_source',
  'properties.bootstrap.servers' = 'Service IP address of the Kafka Broker instance:Kafka port number',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --Delete this parameter if the cluster where FlinkServer resides is in non-security mode.
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --Delete this parameter if the cluster where FlinkServer resides is in non-security mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'  --Delete this parameter if the cluster where FlinkServer resides is in non-security mode.
);
Insert into
  MyUserTable
select
  *
from
  KafkaSource;
```

NOTE

- The Kafka port can be:
  - Value of **sasl.port** when **Authentication Mode** of the cluster is **Security Mode**, **21007** by default.
  - Value of **port** when **Authentication Mode** of the cluster is **Normal Mode**, **9092** by default. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. On the page that is displayed, click the **Configurations** tab then the **All Configurations** sub-tab. On the displayed page, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- **properties.group.id** indicates the Kafka user group ID. This parameter is mandatory when Kafka functions as the source.
- *System domain name*: You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.

Step 3 On the job management page, check whether the job is in the **Running** status.

Step 4 Execute the following commands to view the topic and write data to Kafka. For details, see **Managing Messages in Kafka Topics**.

./kafka-topics.sh --list --zookeeper *Service IP address of the ZooKeeper quorumpeer instance*:*Port number of the ZooKeeper client*/**kafka**

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka instances reside*:*Kafka port number* **--topic** *Topic name* **--producer.config** *Client directory*/**Kafka/kafka/config/producer.properties**

In this example, the topic name is **customer_source**.

sh kafka-console-producer.sh --broker-list *IP address of the node where the Kafka instance is deployed*:*Kafka port number* --**topic customer_source --producer.config /opt/client/Kafka/kafka/config/producer.properties**

Enter the message content.

```
3,zhangsan
4,wangwu
8,zhaosi
```

Press **Enter** to send the message.

NOTE

- Service IP address of the ZooKeeper quorumpeer instance:

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ZooKeeper**. On the page that is displayed, click the **Instance** tab and view the service IP addresses of all nodes where the quorumpeer instances reside.
- Port number of the ZooKeeper client:

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ZooKeeper**. On the page that is displayed, click the **Configurations** tab and check the value of **clientPort**.

Step 5 Log in to the GaussDB client and run the following command to check whether data has been sent to the sink table:

Select * from customer_t1;

```
postgres=> select * from customer_t1;
 c_customer_sk | c_customer_name
---------------+-----------------
             1 | new Data
             8 | zhaosi
             0 | data 0
             3 | zhangsan
             4 | wangwu
             2 | data 2
(6 rows)
```

----End

GaussDB as a Source Table

Step 1 Log in to Manager as user **flinkuser** and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link next to **Flink WebUI** to access the Flink web UI.

Step 2 Create a Flink SQL job by referring to **Creating a FlinkServer Job**. On the job development page, configure the job parameters and start the job.

In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **5000**, and retain the default value for **Mode**.

```
CREATE TABLE MyUserTable(
  --GaussDB functions as a source table.
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH(
  'connector' = 'jdbc',
  'url'='jdbc:gaussdb://IP address of the GaussDB server:Database port number/postgres',
  'table-name' = 'customer_t1',
  'username' = 'username ',
  'password' = 'password '
);
CREATE TABLE KafkaSink (
  -- Kafka functions as a sink table.
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH (
  'connector' = 'kafka',
  'topic' = 'customer_sink',
  'properties.bootstrap.servers' = 'Service IP address of the Kafka Broker instance:Kafka port number',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --Delete this parameter if the cluster where FlinkServer resides is in non-security mode.
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --Delete this parameter if the cluster where FlinkServer resides is in non-security mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'  --Delete this parameter if the cluster where FlinkServer resides is in non-security mode.
);
Insert into
  KafkaSink
select
  *
from
  MyUserTable;
```

NOTE

- The Kafka port can be:
  - Value of **sasl.port** when **Authentication Mode** of the cluster is **Security Mode**, **21007** by default.
  - Value of **port** when **Authentication Mode** of the cluster is **Normal Mode**, **9092** by default. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. On the page that is displayed, click the **Configurations** tab then the **All Configurations** sub-tab. On the displayed page, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- **properties.group.id** indicates the Kafka user group ID. This parameter is mandatory when Kafka functions as the source.
- *System domain name*: You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.

Step 3 On the job management page, check whether the job is in the **Running** status.

Step 4 Run the following command to check whether data is received in the sink table, that is, check whether data is properly written to the Kafka topic. For details, see **Managing Messages in Kafka Topics**.

sh kafka-console-consumer.sh --topic customer_sink --bootstrap-server *IP address of the node where the Kafka instance is deployed:Kafka port number* **--consumer.config /opt/client/Kafka/kafka/config/ consumer.properties**


```
[root@... ... bin]# sh kafka-console-consumer.sh --topic customer_sink --bootstrap-server ...:21007 --consumer.config /opt/client/Kafka/kafka/config/producer.properties
[2022-04-12 11:49:37,957] WARN The configuration 'acks' was supplied but isn't a known config. (org.apache.kafka.clients.consumer.ConsumerConfig)
1,"new Data"
8,zhaosi
4,wangwu
0,"data 0"
3,zhangsan
2,"data 2"
```

----End

GaussDB as a Dimension Table

kafkaSource is used as the fact table, **customer_t2** is used as the dimension table, and the result is written to kafkaSink.

Step 1 Create dimension table **customer_t2** on the GaussDB client. An example of the table creation statement is as follows:

```
CREATE TABLE customer_t2(
c_customer_sk INTEGER PRIMARY KEY,
c_customer_age INTEGER,
c_customer_address VARCHAR(32)
)DISTRIBUTE BY HASH(c_customer_sk);

INSERT INTO customer_t2 VALUES(1,18,'city a');
INSERT INTO customer_t2 VALUES(2,14,'city b');
INSERT INTO customer_t2 VALUES(3,16,'city c');
INSERT INTO customer_t2 VALUES(4,24,'city d');
INSERT INTO customer_t2 VALUES(5,32,'city e');
INSERT INTO customer_t2 VALUES(6,27,'city f');
INSERT INTO customer_t2 VALUES(7,41,'city a');
INSERT INTO customer_t2 VALUES(8,35,'city h');
INSERT INTO customer_t2 VALUES(9,16,'city j');
```

Step 2 Log in to Manager as user **flinkuser** and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link next to **Flink WebUI** to access the Flink web UI.

Step 3 Create a Flink SQL job by referring to **Creating a FlinkServer Job**. On the job development page, configure the job parameters and start the job.

In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **5000**, and retain the default value for **Mode**.

```
CREATE TABLE KafkaSource (
  -- Kafka as a source table
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32),
  proctime as proctime()
) WITH (
  'connector' = 'kafka',
  'topic' = 'customer_source',
  'properties.bootstrap.servers' = 'Service IP address of the Kafka Broker instance:Kafka port number',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --Delete this parameter if the cluster where FlinkServer resides is in non-security mode.
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --Delete this parameter if the cluster where FlinkServer resides is in non-security mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'  --Delete this parameter if the cluster where FlinkServer resides is in non-security mode.
);
CREATE TABLE KafkaSink (
  -- Kafka as a sink table
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32),
  c_customer_age INTEGER,
  c_customer_address VARCHAR(32)
) WITH (
```



```
  'connector' = 'kafka',
  'topic' = 'customer_sink',
  'properties.bootstrap.servers' = 'Service IP address of the Kafka Broker instance:Kafka port number',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --Delete this parameter if the cluster where FlinkServer resides is in non-security mode.
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --Delete this parameter if the cluster where FlinkServer resides is in non-security mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'  --Delete this parameter if the cluster where FlinkServer resides is in non-security mode.
);
CREATE TABLE MyUserTable (
  -- GaussDB as a dimension table
  c_customer_sk INTEGER PRIMARY KEY,
  c_customer_age INTEGER,
  c_customer_address VARCHAR(32)
) WITH (
  'connector' = 'jdbc',
  'url'='jdbc:gaussdb://IP address of the GaussDB server:Database port number/postgres',
  'table-name' = 'customer_t2',
  'username' = 'username',
  'password' = 'password'
);
INSERT INTO
  KafkaSink
SELECT
  t.c_customer_sk,
  t.c_customer_name,
  d.c_customer_age,
  d.c_customer_address
FROM
  KafkaSource as t
  JOIN MyUserTable FOR SYSTEM_TIME AS OF t.proctime as d ON t.c_customer_sk = d.c_customer_sk;
```

NOTE

- The Kafka port can be:
  - Value of **sasl.port** when **Authentication Mode** of the cluster is **Security Mode**, **21007** by default.
  - Value of **port** when **Authentication Mode** of the cluster is **Normal Mode**, **9092** by default. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. On the page that is displayed, click the **Configurations** tab then the **All Configurations** sub-tab. On the displayed page, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- **properties.group.id** indicates the Kafka user group ID. This parameter is mandatory when Kafka functions as the source.
- *System domain name*: You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.

Step 4 Run the following command to check whether data is received in the sink table, that is, check whether data is properly written to the Kafka topic after **Step 5** is performed. For details, see **Managing Messages in Kafka Topics**.

sh kafka-console-consumer.sh --topic customer_sink --bootstrap-server *IP address of the node where the Kafka instance is deployed:Kafka port number* **--consumer.config /opt/client/Kafka/kafka/config/ consumer.properties**

Step 5 View the topic and write data to the Kafka topic by referring to **Managing Messages in Kafka Topics**. After the data is written, view the execution result in the window in **Step 4**.

./kafka-topics.sh --list --zookeeper *Service IP address of the ZooKeeper quorumpeer instance*:*Port number of the ZooKeeper client*/**kafka**

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka instances reside*:*Kafka port number* **--topic** *Topic name* **--producer.config** *Client directory*/**Kafka/kafka/config/producer.properties**

NOTE

- IP address of the ZooKeeper quorumpeer instance

  To obtain the IP addresses of all ZooKeeper quorumpeer instances, log in to FusionInsight Manager and choose **Cluster** > **Services** > **ZooKeeper**. On the displayed page, click **Instance** and view the IP addresses of all the hosts where the quorumpeer instances locate.

- Port number of the ZooKeeper client

  Log in to FusionInsight Manager and choose **Cluster** > **Service** > **ZooKeeper**. On the displayed page, click **Configurations** and check the value of **clientPort**.

In this example, the topic name is **customer_source**.

sh kafka-console-producer.sh --broker-list *IP address of the node where the Kafka instance is deployed*:*Kafka port number* --**topic customer_source --producer.config /opt/client/Kafka/kafka/config/producer.properties**

Enter the message content.

```
3,zhangsan
5,zhaosi
1,xiaoming
2,liuyang
7,liubei
10,guanyu
20,zhaoyun
```

Press **Enter** to send the message. The output in the kafka-console-consumer window in **Step 4** is as follows:

```
3,zhangsan,16,city c
5,zhaosi,32,city e
1,xiaoming,18,city a
2,liuyang,14,city b
7,liubei,41,city a
```

----End

6.5.4 Creating a FlinkServer Job to Interconnect with JDBC

This section applies to MRS 3.3.1 and later.

Scenario

FlinkServer can interconnect with JDBC. This topic uses FlinkServer and Kafka in security mode as an example to describe the DDL definition for JDBC MySQL source tables, sink tables, and dimension tables. This topic also explains the WITH parameters for creating a table and code examples. You will learn how to connect to JDBC on the FlinkServer job management page.

NOTICE

When "FlinkSQL" is displayed in the command output on the FlinkServer web UI in MRS 3.2.0 or later clusters, the **password** field in the SQL statement is left blank. Before you submit a job, manually enter the password.

Mapping between Flink SQL and JDBC data types

Flink SQL	MySQL	Oracle	PostgreSQL	SQL Server
BOOLEAN	BOOLEAN TINYINT(1)	-	BOOLEAN	BIT
TINYINT	TINYINT	-	-	TINYINT
SMALLINT	SMALLINT TINYINT UNSIGNED	-	SMALLINT INT2 SMALLSERIAL SERIAL2	SMALLINT
INT	INT MEDIUMINT SMALLINT UNSIGNED	-	INTEGER SERIAL	INT
BIGINT	BIGINT INT UNSIGNED	-	BIGINT BIGSERIAL	BIGINT
FLOAT	FLOAT	BINARY_FLOAT	REAL FLOAT4	REAL
DOUBLE	DOUBLE DOUBLE PRECISION	BINARY_DOUBLE	FLOAT8 DOUBLE PRECISION	FLOAT
STRING	CHAR(n) VARCHAR(n) TEXT	CHAR(n) VARCHAR(n) CLOB	CHAR(n) CHARACTER(n) VARCHAR(n) CHARACTER VARYING(n) TEXT	CHAR(n) NCHAR(n) VARCHAR(n) NVARCHAR(n) TEXT NTEXT
BYTES	BINARY VARBINARY BLOB	RAW(s) BLOB	BYTEA	BINARY(n) VARBINARY(n)

Flink SQL	MySQL	Oracle	PostgreSQL	SQL Server
ARRAY	-	-	ARRAY	-
DATE	DATE	DATE	DATE	DATE
TIME [(p)] [WITHOUT TIMEZONE]	TIME [(p)]	DATE	TIME [(p)] [WITHOUT TIMEZONE]	TIME(0)
TIMESTAMP [(p)] [WITHOUT TIMEZONE]	DATETIME [(p)]	TIMESTAMP [(p)] [WITHOUT TIMEZONE]	TIMESTAMP [(p)] [WITHOUT TIMEZONE]	DATETIME DATETIME2
DECIMAL(20, 0)	BIGINT UNSIGNED	-	-	-
DECIMAL(p, s)	NUMERIC(p, s) DECIMAL(p, s)	SMALLINT FLOAT(s) DOUBLE PRECISION REAL NUMBER(p, s)	NUMERIC(p, s) DECIMAL(p, s)	DECIMAL(p, s)

Prerequisites

Cluster where FlinkServer resides:

- HDFS, Yarn, Kafka, ZooKeeper, and Flink have been installed in the cluster.
- The client that contains the Kafka service has been installed, for example, in the **/opt/client** directory.
- You have created a user with the **FlinkServer Admin Privilege** (for example, **flinkuser**) for accessing the Flink web UI by referring to **Creating a FlinkServer Role**.

JDBC Sink Table (MySQL as an Example)

Step 1 Create an empty table for receiving data in a database, for example, MySQL database **customer_t1**.

Step 2 Log in to Manager as user **flinkuser** and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link next to **Flink WebUI** to access the Flink web UI.

Step 3 Create a Flink SQL job by referring to **Creating a FlinkServer Job**. On the job development page, configure the job parameters as follows and start the job.

In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **60000**, and retain the default value for **Mode**.

```
CREATE TABLE MyUserTable(
  c_customer_sk INTEGER,
```



```
  c_customer_name VARCHAR(32)
) WITH(
  'connector' = 'jdbc',
  'url' = 'jdbc:mysql://IP address of the MySQL server:MySQL server port/mysql',
  'table-name' = 'customer_t1',
  'username' ='MySQL database username',
  'password' ='Password of the MySQL database user'
);
CREATE TABLE KafkaSource (
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH (
  'connector' = 'kafka',
  'topic' = 'customer_source',
  'properties.bootstrap.servers' = 'Service IP address of the Kafka Broker instance:Kafka port number',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
);
Insert into
  MyUserTable
select
  *
from
  KafkaSource;
```

NOTE

- The IP address and port number of the Kafka broker instance are as follows:
  - To obtain the instance IP address, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, click **Instance**, and query the instance IP address on the instance list page.
  - If Kerberos authentication is enabled for the cluster (the cluster is in security mode), the Broker port number is the value of **sasl.port**. The default value is **21007**.
  - If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), the broker port number is the value of **port**. The default value is **9092**. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. Click **Configurations** then **All Configurations**. On the page that is displayed, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- *System domain name*: To obtain the value, log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.
- **properties.group.id** indicates the Kafka user group ID. This parameter is mandatory when Kafka functions as the source.
- **properties.kerberos.domain.name**: Set it to **hadoop.***System domain name*. To obtain the actual domain name of the cluster, log in to FusionInsight Manager and choose **System** > **Permission** > **Domain and Mutual Trust**.

Step 4 On the job management page, check whether the job status is **Running**.

Step 5 Execute the following commands to view the topic and write data to Kafka. For details, see **Managing Messages in Kafka Topics**.

./kafka-topics.sh --list *--bootstrap-server* *Service IP address of the Kafka Broker instance*:*Kafka port* --command-config *Client directory*/Kafka/kafka/config/client.properties

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka Broker instances reside*:*Kafka port* **--topic** *Topic name* **--producer.config** *Client directory*/Kafka/kafka/config/producer.properties**

In this example, the topic name is **customer_source**.

sh kafka-console-producer.sh --broker-list *IP address of the node where the Kafka Broker instance is located*:*Kafka port* --**topic customer_source --producer.config /opt/client/Kafka/kafka/config/producer.properties**

Enter the message content.

```
3,zhangsan
4,wangwu
8,zhaosi
```

Press **Enter** to send the message.

Step 6 Log in to the MySQL client and run the following statement to check whether the sink table received data:

Select * from customer_t1;

----End

JDBC Source Table (MySQL as an Example)

Step 1 Log in to Manager as user **flinkuser** and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link next to **Flink WebUI** to access the Flink web UI.

Step 2 Create a Flink SQL job by referring to **Creating a FlinkServer Job**. On the job development page, configure the job parameters as follows and start the job.

In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **5000**, and retain the default value for **Mode**.

```
CREATE TABLE MyUserTable(
  --MySQL source table
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH(
  'connector' = 'jdbc',
  'url' = 'jdbc:mysql://IP address of the MySQL server:MySQL server port/mysql',
  'table-name' = 'customer_t1',
  'username' ='MySQL database username',
  'password' ='Password of the MySQL database user'
);
CREATE TABLE KafkaSink (
  -- Kafka functions as a sink table.
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH (
  'connector' = 'kafka',
  'topic' = 'customer_sink',
  'properties.bootstrap.servers' = 'Service IP address of the Kafka Broker instance:Kafka port number',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --Delete this parameter if the cluster where FlinkServer is
```



```
deployed is in non-security mode.
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
);
Insert into
  KafkaSink
select
  *
from
  MyUserTable;
```

NOTE

- The IP address and port number of the Kafka broker instance are as follows:
  - To obtain the instance IP address, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, click **Instance**, and query the instance IP address on the instance list page.
  - If Kerberos authentication is enabled for the cluster (the cluster is in security mode), the Broker port number is the value of **sasl.port**. The default value is **21007**.
  - If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), the broker port number is the value of **port**. The default value is **9092**. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. Click **Configurations** then **All Configurations**. On the page that is displayed, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- *System domain name*: You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.
- **properties.group.id** indicates the Kafka user group ID. This parameter is mandatory when Kafka functions as the source.
- **properties.kerberos.domain.name**: Set it to **hadoop.***System domain name*. You can log in to FusionInsight Manager and choose **System** > **Permission** > **Domain and Mutual Trust** to view the actual domain name of the cluster.

Step 3 On the job management page, check whether the job status is **Running**.

Step 4 Obtain the required Kafka IP address and port by referring to **Managing Messages in Kafka Topics**, and run the following command to check whether data is written from the Kafka topic to the sink table:

sh kafka-console-consumer.sh --topic customer_sink --bootstrap-server *IP address of the node where the Kafka instance is deployed*:*Kafka port number* **--consumer.config /opt/client/Kafka/kafka/config/ consumer.properties**

----End

JDBC Dimension Table (MySQL as an Example)

kafkaSource is used as the fact table, **customer_t2** is used as the dimension table, and the result is written to kafkaSink.

Step 1 Create the dimension table **customer_t2** on the MySQL client. An example of the table creation statement is as follows:

```
CREATE TABLE customer_t2(
c_customer_sk INTEGER PRIMARY KEY,
c_customer_age INTEGER,
c_customer_address VARCHAR(32)
);

INSERT INTO customer_t2 VALUES(1,18,'city a');
INSERT INTO customer_t2 VALUES(2,14,'city b');
INSERT INTO customer_t2 VALUES(3,16,'city c');
INSERT INTO customer_t2 VALUES(4,24,'city d');
INSERT INTO customer_t2 VALUES(5,32,'city e');
INSERT INTO customer_t2 VALUES(6,27,'city f');
INSERT INTO customer_t2 VALUES(7,41,'city a');
INSERT INTO customer_t2 VALUES(8,35,'city h');
INSERT INTO customer_t2 VALUES(9,16,'city j');
```

Step 2 Log in to Manager as user **flinkuser** and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link next to **Flink WebUI** to access the Flink web UI.

Step 3 Create a Flink SQL job by referring to **Creating a FlinkServer Job**. On the job development page, configure the job parameters as follows and start the job.

In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **5000**, and retain the default value for **Mode**.

```
CREATE TABLE KafkaSource (
  -- Kafka as a source table
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32),
  proctime as proctime()
) WITH (
  'connector' = 'kafka',
  'topic' = 'customer_source',
  'properties.bootstrap.servers' = 'Service IP address of the Kafka Broker instance:Kafka port',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
);
CREATE TABLE KafkaSink (
  -- Kafka sink table.
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32),
  c_customer_age INTEGER,
  c_customer_address VARCHAR(32)
) WITH (
  'connector' = 'kafka',
  'topic' = 'customer_sink',
  'properties.bootstrap.servers' = 'Service IP address of the Kafka Broker instance:Kafka port',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
);
CREATE TABLE MyUserTable (
  -- MySQL dimension table
  c_customer_sk INTEGER PRIMARY KEY NOT ENFORCED,
```



```
  c_customer_age INTEGER,
  c_customer_address VARCHAR(32)
) WITH (
  'connector' = 'jdbc',
  'url' = 'jdbc:mysql://IP address of the MySQL server:MySQL server port/mysql',
  'table-name' = 'customer_t2',
  'username' ='MySQL database username',
  'password' ='Password of the MySQL database user'
);
INSERT INTO
  KafkaSink
SELECT
  t.c_customer_sk,
  t.c_customer_name,
  d.c_customer_age,
  d.c_customer_address
FROM
  KafkaSource as t
  JOIN MyUserTable FOR SYSTEM_TIME AS OF t.proctime as d ON t.c_customer_sk = d.c_customer_sk;
```

NOTE

- The IP address and port number of the Kafka broker instance are as follows:
  - To obtain the instance IP address, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, click **Instance**, and query the instance IP address on the instance list page.
  - If Kerberos authentication is enabled for the cluster (the cluster is in security mode), the Broker port number is the value of **sasl.port**. The default value is **21007**.
  - If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), the broker port number is the value of **port**. The default value is **9092**. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. Click **Configurations** then **All Configurations**. On the page that is displayed, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- *System domain name*: You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.
- **properties.group.id** indicates the Kafka user group ID. This parameter is mandatory when Kafka functions as the source.
- **properties.kerberos.domain.name**: Set it to **hadoop.***System domain name*. You can log in to FusionInsight Manager and choose **System** > **Permission** > **Domain and Mutual Trust** to view the actual domain name of the cluster.

Step 4 Run the following command to check whether data is received in the sink table, that is, check whether data is properly written to the Kafka topic after **Step 5** is performed. For details, see **Managing Messages in Kafka Topics**.

sh kafka-console-consumer.sh --topic customer_sink --bootstrap-server *IP address of the node where the Kafka instance is deployed*:*Kafka port number* **--consumer.config /opt/client/Kafka/kafka/config/ consumer.properties**

Step 5 View the topic and write data to the Kafka topic by referring to **Managing Messages in Kafka Topics**. After the data is written, view the execution result in the window in **Step 4**.

./kafka-topics.sh --list *Service IP address of the Kafka Broker instance*:*Kafka port* **--command-config** *Client directory*/**Kafka/kafka/config/client.properties**

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka Broker instances are deployed*:*Kafka port* **--topic** *Topic name* **--producer.config** *Client directory*/**Kafka/kafka/config/producer.properties**

In this example, the topic name is **customer_source**.

sh kafka-console-producer.sh --broker-list *IP address of the node where the Kafka Broker instance is deployed*:*Kafka port* **--topic customer_source --producer.config /opt/client/Kafka/kafka/config/producer.properties**

Enter the message content.

```
3,zhangsan
5,zhaosi
1,xiaoming
2,liuyang
7,liubei
10,guanyu
20,zhaoyun
```

Press **Enter** to send the message. The output in the kafka-console-consumer window in **Step 4** is as follows:

```
3,zhangsan,16,city c
5,zhaosi,32,city e
1,xiaoming,18,city a
2,liuyang,14,city b
7,liubei,41,city a
```

----End

6.5.5 Creating a FlinkServer Job to Write Data to a DWS

This section applies to MRS 3.3.1 and later.

Scenario

Data Warehouse Service (DWS) is an online data analysis and processing database. This topic uses the FlinkServer and Kafka in security mode, and takes DWS as the sink table to explain the "with" parameters for creating a table and code examples. You will learn how to interconnect with DWS on the FlinkServer job management page.

NOTICE

When "FlinkSQL" is displayed in the command output on the FlinkServer web UI in MRS 3.2.0 or later clusters, the **password** field in the SQL statement is left blank. Before you submit a job, manually enter the password.

Prerequisites

In the cluster where FlinkServer is deployed:

- HDFS, Yarn, Kafka, ZooKeeper, and Flink have been installed.
- The client that contains the Kafka service has been installed, for example, in the **/opt/client** directory.
- You have created a user with the **FlinkServer Admin Privilege** (for example, **flinkuser**) for accessing the Flink web UI by referring to **Creating a FlinkServer Role**.

Procedure

Step 1 Create an empty table for receiving data in a DWS database, for example, **dws_test**.

Step 2 Log in to Manager as user **flinkuser** and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link next to **Flink WebUI** to access the Flink web UI.

Step 3 Create a Flink SQL job by referring to **Creating a FlinkServer Job**. On the job development page, configure the job parameters as follows and start the job.

In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **60000**, and retain the default value for **Mode**.

```
CREATE TABLE dws_test(
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH(
  'connector' = 'dws',
  'url' = 'jdbc:postgresql://DWS connection address/gaussdb',
  'table-name' = 'dws_test',
  'username' ='DWS username',
  'password' ='DWS user password'


);
CREATE TABLE KafkaSource (
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH (
  'connector' = 'kafka',
  'topic' = 'customer_source',
  'properties.bootstrap.servers' = 'Service IP address of the Kafka Broker instance:Kafka port',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'  --Delete this parameter if the cluster where FlinkServer is deployed is in non-security mode.
);
Insert into
  dws_test
select
  *
from
  KafkaSource;
```


NOTE

- The IP address and port number of the Kafka broker instance are as follows:
  - To obtain the instance IP address, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, click **Instance**, and query the instance IP address on the instance list page.
  - If Kerberos authentication is enabled for the cluster (the cluster is in security mode), the Broker port number is the value of **sasl.port**. The default value is **21007**.
  - If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), the broker port number is the value of **port**. The default value is **9092**. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. Click **Configurations** then **All Configurations**. On the page that is displayed, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- *System domain name*: To obtain the value, log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.
- **properties.group.id** indicates the Kafka user group ID. This parameter is mandatory when Kafka functions as the source.
- **properties.kerberos.domain.name**: Set it to **hadoop.***System domain name*. To obtain the actual domain name of the cluster, log in to FusionInsight Manager and choose **System** > **Permission** > **Domain and Mutual Trust**.
- The DWS sink supports the configuration of batch parameters. When one of the conditions is met, the sink is triggered.
  - **autoFlushBatchSize**: size of the batch for automatically refreshing the database. Default value: **5000**.
  - **autoFlushMaxInterval**: maximum interval for automatically refreshing the database (duration to form a batch). Default value: **5s**.

Step 4 On the job management page, check whether the job status is **Running**.

Step 5 Execute the following commands to view the topic and write data to Kafka. For details, see **Managing Messages in Kafka Topics**.

./kafka-topics.sh --list *--bootstrap-server* *Service IP address of the Kafka Broker instance*:*Kafka port* **--command-config** *Client directory*/**Kafka/kafka/config/client.properties**

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka Broker instances reside*:*Kafka port* **--topic** *Topic name* **--producer.config** *Client directory*/**Kafka/kafka/config/producer.properties**

In this example, the topic name is **customer_source**.

sh kafka-console-producer.sh --broker-list *IP address of the node where the Kafka Broker instance is located*:*Kafka port* --**topic customer_source --producer.config /opt/client/Kafka/kafka/config/producer.properties**

Enter the message content.

```
3,zhangsan
4,wangwu
8,zhaosi
```

Press **Enter** to send the message.

Step 6 Log in to the DWS client and run the following command to check whether the sink table received data:

Select * from dws_test;

----End

6.5.6 Creating a FlinkServer Job to Write Data to an HBase Table

This section applies to MRS 3.1.2 or later clusters.

Scenario

FlinkServer can be interconnected with HBase. The details are as follows:

- It can be interconnected with dimension tables and sink tables.
- When HBase and Flink are in the same cluster or clusters with mutual trust, FlinkServer can be interconnected with HBase.
- If HBase and Flink are in different clusters without mutual trust, Flink in a normal cluster can be interconnected with HBase in a normal cluster.

Prerequisites

- The HDFS, Yarn, Flink, and HBase services have been installed in a cluster.
- The client that contains the HBase service has been installed, for example, in the **/opt/client** directory.

Creating a Job

Scenario 1: HBase Functioning as a Sink Table

Step 1 Create a table on the HBase client.

Log in to the HBase client by referring to **Using the HBase Client** and run the **create'dim_province', "f1"** command to create the **dim_province** table.

Step 2 Copy the HBase configuration file to the node where FlinkServer is deployed.

1. Log in to the node where the client is installed as the client installation user and copy all configuration files in the **/opt/client/HBase/hbase/conf/** directory of HBase to an empty directory of all nodes where FlinkServer is deployed, for example, **/tmp/client/HBase/hbase/conf/**.
2. Change the owner of the configuration file directory and its upper-layer directory on the FlinkServer node to **omm**.

   chown omm: /tmp/client/HBase/ -R

   NOTE

   - FlinkServer nodes:

     Log in to Manager, choose **Cluster** > **Services** > **Flink** > **Instance**, and check the **Service IP Address** of FlinkServer.
   - If the node where a FlinkServer instance is located is the node where the HBase client is installed, skip this step on this node.

Step 3 Add the local path for FlinkServer to access the HBase cluster.

Log in to Manager and choose **Cluster** > **Services** > **Flink**. Click **Configurations** then **All Configurations**, search for the **HBASE_CONF_DIR** parameter, and enter the FlinkServer directory (for example, **/tmp/client/HBase/hbase/conf/**) to which the HBase configuration files are copied from **Value**. After the parameters are configured, click **Save**. After confirming the modification, click **OK**.

NOTE

If the node where a FlinkServer instance is deployed is the node where the HBase client is installed, enter the **/opt/client/HBase/hbase/conf/** directory of HBase in **Value** of the **HBASE_CONF_DIR** parameter.

Step 4 Restart the affected FlinkServer instance.

Click **Instance**, select all FlinkServer instances, choose **More** > **Restart Instance**, enter the password, and click **OK** to restart the instances.

Step 5 Access FlinkServer and create a Flink SQL job.

1. Log in to Manager and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link on the right of **Flink WebUI** to access the Flink web UI.
2. Create a Flink SQL job and set Task Type to Stream job. For details, see **Creating a FlinkServer Job**. On the job development page, configure the job parameters as follows and start the job.

   In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **60000**, and retain the default value for **Mode**.

   If the cluster is in security mode and the HBase authentication setting is **hbase.rpc.protection=authentication**, create a Flink SQL job by referring to the following example:

```
CREATE TABLE ksource1 (
user_id STRING,
item_id STRING,
proctime as PROCTIME()
) WITH (
'connector' = 'kafka',
'topic' = 'ksource1',
'properties.group.id' = 'group1',
'properties.bootstrap.servers' = 'IP address of the Kafka broker instance 1:Kafka port number,IP address of the Kafka broker instance 2:Kafka port number',
'format' = 'json',
'properties.sasl.kerberos.service.name' = 'kafka',--This parameter is not required for clusters in normal mode.
'properties.security.protocol' = 'SASL_PLAINTEXT',--This parameter is not required for clusters in normal mode.
'properties.kerberos.domain.name' = 'hadoop.System domain name'--This parameter is not required for clusters in normal mode.
);

CREATE TABLE hsink1 (
rowkey STRING,
f1 ROW < item_id STRING >,
PRIMARY KEY (rowkey) NOT ENFORCED
) WITH (
'connector' = 'hbase-2.2',
'table-name' = 'dim_province',
'zookeeper.quorum' = 'IP address of the ZooKeeper quorumpeer instance 1:ZooKeeper port number,IP address of the ZooKeeper quorumpeer instance 2:ZooKeeper port number'
);

INSERT INTO
```



```
hsink1
SELECT
user_id as rowkey,
ROW(item_id) as f1
FROM
ksource1;
```

NOTE

- The IP address and port number of the Kafka broker instance are as follows:
  - To obtain the instance IP address, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, click **Instance**, and query the instance IP address on the instance list page.
  - If Kerberos authentication is enabled for the cluster (the cluster is in security mode), the Broker port number is the value of **sasl.port**. The default value is **21007**.
  - If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), the broker port number is the value of **port**. The default value is **9092**. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. Click **Configurations** then **All Configurations**. On the page that is displayed, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- *System domain name*: You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.
- IP address of the ZooKeeper quorumpeer instance

  To obtain IP addresses of all ZooKeeper quorumpeer instances, log in to FusionInsight Manager and choose **Cluster** > **Services** > **ZooKeeper**. On the displayed page, click **Instance** and view the IP addresses of all the hosts where the quorumpeer instances locate.
- Port number of the ZooKeeper client

  Log in to FusionInsight Manager and choose **Cluster** > **Service** > **ZooKeeper**. On the displayed page, click **Configurations** and check the value of **clientPort**.
- HBase authentication

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HBase**, click **Configuration** and then **All Configurations**, search for **hbase.rpc.protection**, and check the HBase authentication mode. If the authentication mode is **integrity** or **privacy**, add the following parameters:

  'properties.hbase.rpc.protection' = '*HBase authentication mode*'

  'properties.zookeeper.znode.parent' = '/hbase'

  'properties.hbase.security.authorization' = 'true'

  'properties.hbase.security.authentication' = 'kerberos'

3. On the job management page, check whether the job status is **Running**.

Step 6 Execute the following script to write data to Kafka. For details, see **Managing Messages in Kafka Topics**.

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka Broker instance is deployed*:*Kafka port* **--topic** *Topic name* **--producer.config** *Client directory*/**Kafka/kafka/config/producer.properties**

For example, if the topic name is **ksource1**, the script is **sh kafka-console-producer.sh --broker-list** *IP address of the node where the Kafka Broker instance is located*:*Kafka port* **--topic ksource1 --producer.config /opt/client/Kafka/kafka/config/producer.properties**

Enter the message content and press **Enter** to send the message.

```
{"user_id": "3","item_id":"333333"}
{"user_id": "4","item_id":"44444444"}
```

Step 7 Log in to the HBase client and view the table data.

hbase shell

scan 'dim_province'

----End

Scenario 2: HBase Functioning as a Dimension Table

Step 1 Create a table on the HBase client and write data into the table.

Log in to the HBase client, use **create 'hbase_dim_table','f1"** to create the **hbase_dim_table** table, and write data into the table. For details, see **Using the HBase Client**.

```
put 'hbase_dim_table','1','f1:address','city1'
put 'hbase_dim_table','2','f1:address','city2'
put 'hbase_dim_table','3','f1:address','city3'
```

Step 2 Copy the HBase configuration file to the node where FlinkServer is deployed.

1. Log in to the node where the client is installed as the client installation user and copy all configuration files in the **/opt/client/HBase/hbase/conf/** directory of HBase to an empty directory of all nodes where FlinkServer is deployed, for example, **/tmp1/client/HBase/hbase/conf/**.
2. Change the owner of the configuration file directory and its upper-layer directory on the FlinkServer node to **omm**.

   chown omm: /tmp1/client/HBase/ -R

   NOTE

   - FlinkServer nodes:

     Log in to Manager, choose **Cluster** > **Services** > **Flink** > **Instance**, and check the **Service IP Address** of FlinkServer.
   - If the node where a FlinkServer instance is located is the node where the HBase client is installed, skip this step on this node.

Step 3 Add the local path for FlinkServer to access the HBase cluster.

Log in to Manager and choose **Cluster** > **Services** > **Flink**. Click **Configurations** then **All Configurations**, search for the **HBASE_CONF_DIR** parameter, and enter the FlinkServer directory (for example, **/tmp1/client/HBase/hbase/conf/**) to which the HBase configuration files are copied from **Value**. After the parameters are configured, click **Save**. After confirming the modification, click **OK**.

NOTE

If the node where a FlinkServer instance is deployed is the node where the HBase client is installed, enter the **/opt/client/HBase/hbase/conf/** directory of HBase in **Value** of the **HBASE_CONF_DIR** parameter.

Step 4 Restart the affected FlinkServer instance.

Click **Instance**, select all FlinkServer instances, choose **More** > **Restart Instance**, enter the password, and click **OK** to restart the instances.

Step 5 Access FlinkServer and create a Flink SQL job.

1. Log in to Manager and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link on the right of **Flink WebUI** to access the Flink web UI.
2. Create a Flink SQL job and set Task Type to Stream job. For details, see **Creating a FlinkServer Job**. On the job development page, configure the job parameters as follows and start the job.

   In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **60000**, and retain the default value for **Mode**.

   If the cluster is in security mode and the HBase authentication setting is **hbase.rpc.protection=authentication**, create a Flink SQL job by referring to the following example:

```
CREATE TABLE KafkaSource (
  `user_id` STRING,
  `user_name` STRING,
  proctime as proctime()
) WITH (
  'connector' = 'kafka',
  'topic' = 'user_source',
  'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',--This parameter is not required for clusters in normal mode.
'properties.security.protocol' = 'SASL_PLAINTEXT',--This parameter is not required for clusters in normal mode.
'properties.kerberos.domain.name' = 'hadoop.System domain name'--This parameter is not required for clusters in normal mode.
);
CREATE TABLE KafkaSink (
  -- Kafka functions as a sink table.
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `address` VARCHAR
) WITH (
  'connector' = 'kafka',
  'topic' = 'user_sink',
  'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',--This parameter is not required for clusters in normal mode.
'properties.security.protocol' = 'SASL_PLAINTEXT',--This parameter is not required for clusters in normal mode.
'properties.kerberos.domain.name' = 'hadoop.System domain name'--This parameter is not required for clusters in normal mode.
);
CREATE TABLE hbaseTable (
  -- HBase dimension table
  user_id STRING,
  f1 ROW < address STRING >,
  PRIMARY KEY (user_id) NOT ENFORCED
) WITH (
  'connector' = 'hbase-2.2',
  'table-name' = 'hbase_dim_table',
  'zookeeper.quorum' = 'IP address of the ZooKeeper quorumpeer instance 1:ZooKeeper port number,IP address of the ZooKeeper quorumpeer instance 2:ZooKeeper port number'
);
INSERT INTO
  KafkaSink
SELECT
  t.user_id,
```



```
  t.user_name,
  d.address
FROM
  KafkaSource as t
  JOIN hbaseTable FOR SYSTEM_TIME AS OF t.proctime as d ON t.user_id = d.user_id;
```

3. On the job management page, check whether the job status is **Running**.

Step 6 Execute the following script to write data to Kafka. For details, see **Managing Messages in Kafka Topics**.

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka Broker instance is deployed*:*Kafka port* **--topic** *Topic name* **--producer.config** *Client directory*/**Kafka/kafka/config/producer.properties**

Enter the message content and press **Enter** to send the message.

```
1,name1
2,name2
3,name3
```

Step 7 Run the following commands to check whether data is written from the Kafka topic to the sink table:

sh kafka-console-consumer.sh --topic *Topic name* **--bootstrap-server** *Service IP address of the Kafka broker instance*:*Kafka port number* --**consumer.config** *Client directory*/**Kafka/kafka/config/consumer.properties**

The result is as follows:

```
1,name1,city1
2,name2,city2
3,name3,city3
```

----End

Submitting a Job Using the Application

- If the Flink run mode is used, you are advised to use the **export HBASE_CONF_DIR=** *HBase configuration directory*, for example, **export HBASE_CONF_DIR=/opt/hbaseconf**.
- If the Flink run-application mode is used, you can use either of the following methods to submit jobs:
  - (Recommended) Add the following configurations to a table creation statement.

Table 6-16 Related configurations

Parameter	Description
'properties.hbase.rpc.protection' = 'authentication'	This parameter must be consistent with that on the HBase server.
'properties.zookeeper.znode.parent' = '/hbase'	If there are multiple services, hbase1 and hbase2 coexist. You must clarify the cluster to be accessed.
'properties.hbase.security.authorization' = 'true'	Authentication is enabled.

Parameter	Description
'properties.hbase.security.authentication' = 'kerberos'	Kerberos authentication is enabled.

Example:

```
CREATE TABLE hsink1 (
    rowkey STRING,
    f1 ROW < q1 STRING >,
    PRIMARY KEY (rowkey) NOT ENFORCED
  ) WITH (
    'connector' = 'hbase-2.2',
    'table-name' = 'cc',
    'zookeeper.quorum' = 'x.x.x.x:clientPort',
    'properties.hbase.rpc.protection' = 'authentication',
    'properties.zookeeper.znode.parent' = '/hbase',
    'properties.hbase.security.authorization' = 'true',
    'properties.hbase.security.authentication' = 'kerberos'
  );
```

- Add the HBase configuration to YarnShip.

  Example: Dyarn.ship-files=/opt/hbaseconf

6.5.7 Creating a FlinkServer Job to Write Data to an HDFS

This section applies to MRS 3.1.2 or later clusters.

Scenario

This section describes the data definition language (DDL) of HDFS as a sink table, as well as the WITH parameters and example code for creating a sink table, and provides guidance on how to perform operations on the FlinkServer job management page.

If your Kafka cluster is in security mode, the following example SQL statements can be used.

Prerequisites

- The HDFS, Yarn, and Flink services have been installed in a cluster.
- The client that contains the HDFS service has been installed, for example, in the **/opt/client** directory.
- You have created a user assigned with the **FlinkServer Admin Privilege** (for example, **flink_admin**) for accessing the Flink web UI by referring to **Creating a FlinkServer Role**.

Creating a Job

Step 1 Log in to Manager as user **flink_admin** and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link on the right of **Flink WebUI** to access the Flink web UI.

Step 2 Create a Flink SQL job by referring to **Creating a FlinkServer Job**. On the job development page, configure the job parameters and start the job.

In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **60000**, and retain the default value for **Mode**.

```
CREATE TABLE kafka_table (
  user_id STRING,
  order_amount DOUBLE,
  log_ts TIMESTAMP(3),
  WATERMARK FOR log_ts AS log_ts - INTERVAL '5' SECOND
) WITH (
  'connector' = 'kafka',
  'topic' = 'user_source',
  'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port number',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'format' = 'csv',
   --Ignore the CSV data that fails to be parsed.
  'csv.ignore-parse-errors' = 'true',--If the data is in JSON format, set json.ignore-parse-errors to true.
  'properties.sasl.kerberos.service.name' = 'kafka',--This parameter is not required for clusters in normal mode. Delete the comma (,) in the previous line.
  'properties.security.protocol' = 'SASL_PLAINTEXT',--This parameter is not required for clusters in normal mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'--This parameter is not required for clusters in normal mode.

);

CREATE TABLE fs_table (
  user_id STRING,
  order_amount DOUBLE,
  dt STRING,
  `hour` STRING
) PARTITIONED BY (dt, `hour`) WITH ( --Date-specific file partitioning
  'connector'='filesystem',
  'path'='hdfs:///sql/parquet',
  'format'='parquet',
  'sink.partition-commit.delay'='0 s',-- Partitions will not be committed before the delay time. If the files are partitioned by day, set this parameter to '1 d'. If the files are partitioned by hour, set this parameter to '1 h'.
  'sink.partition-commit.policy.kind'='success-file'
);
-- streaming sql, insert into file system table
INSERT INTO fs_table SELECT user_id, order_amount, DATE_FORMAT(log_ts, 'yyyy-MM-dd'), DATE_FORMAT(log_ts, 'HH') FROM kafka_table;
```

NOTE

- Kafka port
  - Value of **sasl.port** when **Authentication Mode** of the cluster is **Security Mode**, **21007** by default.
  - Value of **port** when **Authentication Mode** of the cluster is **Normal Mode**, **9092** by default. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. On the page that is displayed, click the **Configurations** tab then the **All Configurations** sub-tab. On the displayed page, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- *System domain name*: You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.

Step 3 On the job management page, check whether the job status is **Running**.

Step 4 Execute the following commands to view the topic and write data to Kafka. For details, see **Managing Messages in Kafka Topics**.

./kafka-topics.sh --list *--bootstrap-server* *Service IP address of the Kafka Broker instance:Kafka port* **--command-config** *Client directory*/**Kafka/kafka/config/client.properties**

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka instances reside*:*Kafka port number* **--topic** *Topic name* **--producer.config** *Client directory*/**Kafka/kafka/config/producer.properties**

For example, if the topic name is **user_source**, the script is **sh kafka-console-producer.sh --broker-list** *IP address of the node where Kafka instances are deployed*:*Kafka port* **--topic user_source** --**producer.config /opt/client/Kafka/kafka/config/producer.properties**

Enter the message content.

```
3,3333,"2021-09-10 14:00"
4,4444,"2021-09-10 14:01"
```

Press **Enter** to send the message.

Step 5 Run the following command to check whether data is written from the HDFS directory to the sink table:

hdfs dfs -ls -R /sql/parquet

----End

Interconnecting Flink with HDFS Partitions

- Customized partitioning

  Flink's file system supports partitions in the standard Hive format. You do not need to register partitions with a table catalog. Partitions are inferred based on the directory structure.

  For example, a table that is partitioned based on the following directory is inferred to contain datetime and hour partitions.

```
path
└── datetime=2021-09-03
    └── hour=11
        ├── part-0.parquet
        ├── part-1.parquet
    └── hour=12
        ├── part-0.parquet
└── datetime=2021-09-24
    └── hour=6
        ├── part-0.parquet
```

- Rolling policy of partition files

  Data in the partition directories is split into part files. Each partition contains at least one part file, which is used to receive the data written by the subtask of the sink.

  The following parameters describe the rolling policies of partition files.

  Table 6-17 Rolling policy of partition files

Parameter	Default Value	Type	Description
sink.rolling-policy.file-size	128 MB	Memory Size	Maximum size of a partition file before it is rolled.

Parameter	Default Value	Type	Description
sink.rolling-policy.rollover-interval	30 minutes	Duration	Maximum duration that a partition file can stay open before it is rolled.
sink.rolling-policy.check-interval	1 minute	Duration	Interval for checking time-based rolling policies.

- File merging

  File compression is supported, allowing applications to have a shorter checkpoint interval without generating a large number of files.

  NOTE

  Only files in a single checkpoint are compressed. That is, the number of generated files is at least the same as the number of checkpoints. Files are invisible before merged. They are visible after both the checkpoint and compression are complete. If file compression takes too much time, the checkpoint will be prolonged.

  Table 6-18 File merging in partition directories

Parameter	Default Value	Type	Description
auto-compaction	false	Boolean	Whether to enable automatic compression. Data will be written to temporary files. After a checkpoint is complete, the temporary files generated by the checkpoint are compressed. These temporary files are invisible before compression.
compaction.file-size	none	Memory Size	Size of the target file to be compressed. The default value is the size of the file to be rolled.

- Partition commit

  After a file is written to a partition, for example, a partition is added to Hive metastore (HMS) or a **_SUCCESS** file is written to a directory, the downstream application needs to be notified. Triggers and policies are used to commit partition files.

Table 6-19 Trigger parameters for committing partition files

Parameter	Default Value	Type	Description
sink.partition-commit.trigger	process-time	String	• process-time: System time of the compute node. It does not need to extract the partition time or generate watermarks. If the current system time exceeds the system time generated when a partition is created plus the delay time, the partition should be submitted. • partition-time: Time extracted from the partition. Watermarks are required. If the time for generating watermarks exceeds the time extracted from a partition plus the delay time, the partition should be submitted.
sink.partition-commit.delay	0 s	Duration	Partitions will not be committed before the delay time. If it is a daily partition, the value is **1 d**. If it is an hourly one, the value is **1 h**.

Table 6-20 Policy parameters for committing partition files

Configuration Item	Default Value	Type	Description
sink.partition-commit.policy.kind	-	String	Policy for committing partitions: • **metastore**: used to add partitions to metastore. Only Hive tables support the metastore policy. The file system manages partitions based on the directory structure. • **success-file**: used to add **success-file** files to a directory. • The two policies can be configured at the same time, that is, **'sink.partition-commit.policy.kind'='metastore,success-file'**.
sink.partition-commit.policy.class	-	String	Class that implements partition commit policy interfaces. This parameter takes effect only in the customized submission policies.

Configuration Item	Default Value	Type	Description
sink.partition-commit.success-file.name	_SUCCESS	String	File name of the success-file partition commit policy. The default value is **_SUCCESS**.

6.5.8 Creating a FlinkServer Job to Write Data to a Hive Table

This section applies to MRS 3.1.2 or later clusters.

Scenario

Currently, FlinkServer interconnects with Hive MetaStore. Therefore, the MetaStore function must be enabled for Hive. Hive can be used as sink, and dimension tables.

If your Kafka cluster is in security mode, the following example SQL statements can be used.

Prerequisites

- Services such as HDFS, Yarn, Kafka, Flink, and Hive (the service name must be **Hive**) have been installed in the cluster.
- The client that contains the Hive service has been installed, for example, in the **/opt/client** directory.
- Flink 1.12.2 or later and Hive 3.1.0 or later are supported.
- You have created a user assigned with the **FlinkServer Admin Privilege** (for example, **flink_admin**) for accessing the Flink web UI by referring to **Creating a FlinkServer Role**.
- You have obtained the client configuration file and credential of the user for accessing the Flink web UI. For details, see "Note" in **Procedure**.

Creating a Job

Scenario 1: Hive Functioning as a Sink Table

The following uses the process of interconnecting a Kafka mapping table to Hive as an example.

Step 1 Log in to the Flink web UI as user **flink_admin**. For details, see **Accessing the FlinkServer Web UI**.

Step 2 Create a cluster connection, for example, **flink_hive**.

1. Choose **System Management** > **Cluster Connection Management**. The **Cluster Connection Management** page is displayed.
2. Click **Create Cluster Connection**. On the displayed page, enter information by referring to **Table 6-21** and click **Test**. After the test is successful, click **OK**.

Table 6-21 Parameters for creating a cluster connection

Parameter	Description	Example Value
Cluster Connection Name	Name of the cluster connection, which can contain a maximum of 100 characters. Only letters, digits, and underscores (_) are allowed.	flink_hive
Description	Description of the cluster connection name.	-
Version	Select a cluster version.	MRS 3
Secure Version	– If the secure version is used, select **Yes** for a security cluster. Enter the username and upload the user credential. – If not, select **No**.	Yes
Username	The user must have the minimum permissions for accessing services in the cluster. The name can contain a maximum of 100 characters. Only letters, digits, and underscores (_) are allowed. This parameter is available only when **Secure Version** is set to **Yes**.	flink_admin
Client Profile	Client profile of the cluster, in TAR format.	-
User Credential	User authentication credential in FusionInsight Manager in TAR format. This parameter is available only when **Secure Version** is set to **Yes**. Files can be uploaded only after the username is entered.	User credential of **flink_admin**

Step 3 Create a Flink SQL job, for example, **flinktest1**.

1. Click **Job Management**. The job management page is displayed.
2. Click **Create Job**. On the displayed job creation page, set parameters by referring to **Table 6-22** and click **OK**. The job development page is displayed.

Table 6-22 Parameters for creating a job

Parameter	Description	Example Value
Type	Job type, which can be **Flink SQL** or **Flink Jar**.	Flink SQL
Name	Job name, which can contain a maximum of 64 characters. Only letters, digits, and underscores (_) are allowed.	flinktest1

Parameter	Description	Example Value
Task Type	Type of the job data source, which can be a stream job or a batch job.	Stream job
Description	Job description, which can contain a maximum of 100 characters.	-

Step 4 On the job development page, enter the following statements and click **Check Semantic** to check the input content.

```
CREATE TABLE test_kafka (
  user_id varchar,
  item_id varchar,
  cat_id varchar,
  zw_test timestamp
) WITH (
  'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port number',
  'format' = 'json',
  'topic' = 'zw_tset_kafka',
  'connector' = 'kafka',
  'scan.startup.mode' = 'latest-offset',
  'properties.sasl.kerberos.service.name' = 'kafka',--This parameter is not required for clusters in normal mode. Delete the comma (,) in the previous line.
  'properties.security.protocol' = 'SASL_PLAINTEXT',--This parameter is not required for clusters in normal mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'--This parameter is not required for clusters in normal mode.

);
CREATE CATALOG myhive WITH (
  'type' = 'hive',
  'hive-version' = '3.1.0',
  'default-database' = 'default',
  'cluster.name' = 'flink_hive'
);
use catalog myhive;
set table.sql-dialect = hive;create table user_behavior_hive_tbl (
    user_id STRING,
    item_id STRING,
    cat_id STRING,
    ts timestamp
  ) PARTITIONED BY (dy STRING, ho STRING, mi STRING) stored as textfile TBLPROPERTIES (
    'partition.time-extractor.timestamp-pattern' = '$dy $ho:$mi:00',
    'sink.partition-commit.trigger' = 'process-time',
    'sink.partition-commit.delay' = '0S',
    'sink.partition-commit.policy.kind' = 'metastore,success-file'
  );
INSERT into
  user_behavior_hive_tbl
SELECT
  user_id,
  item_id,
  cat_id,
  zw_test,
  DATE_FORMAT(zw_test, 'yyyy-MM-dd'),
  DATE_FORMAT(zw_test, 'HH'),
  DATE_FORMAT(zw_test, 'mm')
FROM
  default_catalog.default_database.test_kafka;
```


NOTE

- Kafka port
  - Value of **sasl.port** when **Authentication Mode** of the cluster is **Security Mode**, **21007** by default.
  - Value of **port** when **Authentication Mode** of the cluster is **Normal Mode**, **9092** by default. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. On the page that is displayed, click the **Configurations** tab then the **All Configurations** sub-tab. On the displayed page, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- The value of **'cluster.name'** is the name of the cluster connection created in **Step 2**.
- *System domain name*: You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.

Step 5 After the job is developed, in **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **60000**, and retain the default value for **Mode**.

Step 6 Click **Submit** in the upper left corner to submit the job.

Step 7 After the job is successfully executed, choose **More** > **Job Monitoring** to view the job running details.

Step 8 Execute the following commands to view the topic and write data to Kafka. For details, see **Managing Messages in Kafka Topics**.

./kafka-topics.sh --list *--bootstrap-server* *Service IP address of the Kafka Broker instance*:*Kafka port* **--command-config** *Client directory*/**Kafka/kafka/config/client.properties**

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka instances reside*:*Kafka port number* **--topic** *Topic name* **--producer.config** *Client directory*/**Kafka/kafka/config/producer.properties**

For example, if the topic name is **zw_tset_kafka**, the script is **sh kafka-console-producer.sh --broker-list** *IP address of the node where Kafka instances are deployed*:*Kafka port* **--topic zw_tset_kafka** --**producer.config /opt/client/Kafka/kafka/config/producer.properties**

Enter the message content.

```
{"user_id": "3","item_id":"333333","cat_id":"cat333","zw_test":"2021-09-08 09:08:01"}
{"user_id": "4","item_id":"444444","cat_id":"cat444","zw_test":"2021-09-08 09:08:01"}
```

Press **Enter** to send the message.

Step 9 Run the following command to check whether data is written from the Hive table to the sink table:

beeline

select * from user_behavior_hive_tbl;

----End

Scenario 2: Hive Functioning as a Dimension Table

Step 1 Log in to the Hive client by referring to **Using the Hive Client**, create a Hive table, and insert data.


```
CREATE TABLE hive3 ( id  int, name string ) PARTITIONED BY (dy STRING,ho STRING,mi STRING)
STORED AS textfile TBLPROPERTIES (
'partition.time-extractor.timestamp-pattern'='$dy $ho:$mi:00',
'streaming-source.partition.include'='all',
'streaming-source.enable'='true',
'streaming-source.partition.include'='all',
'streaming-source.monitor-interval'='2m');

insert into table hive3  values('1','aname','sss','name1','company1');
insert into table hive3  values('2','bname','sss','name1','company1');
```

Step 2 Log in to the Flink web UI as user **flink_admin**. For details, see **Accessing the FlinkServer Web UI**.

Step 3 Create a cluster connection, for example, **flink_hive1**.

1. Choose **System Management** > **Cluster Connection Management**. The **Cluster Connection Management** page is displayed.
2. Click **Create Cluster Connection**. On the displayed page, enter information by referring to **Table 6-23** and click **Test**. After the test is successful, click **OK**.

Table 6-23 Parameters for creating a cluster connection

Parameter	Description	Example Value
Cluster Connection Name	Name of the cluster connection, which can contain a maximum of 100 characters. Only letters, digits, and underscores (_) are allowed.	flink_hive1
Description	Description of the cluster connection name.	-
Version	Select a cluster version.	MRS 3
Secure Version	– If the secure version is used, select **Yes** for a security cluster. Enter the username and upload the user credential. – If not, select **No**.	Yes
Username	The user must have the minimum permissions for accessing services in the cluster. The name can contain a maximum of 100 characters. Only letters, digits, and underscores (_) are allowed. This parameter is available only when **Secure Version** is set to **Yes**.	flink_admin
Client Profile	Client profile of the cluster, in TAR format.	-
User Credential	User authentication credential in FusionInsight Manager in TAR format. This parameter is available only when **Secure Version** is set to **Yes**. Files can be uploaded only after the username is entered.	User credential of **flink_admin**

Step 4 Create a Flink SQL job, for example, **flinktest2**.

1. Click **Job Management**. The job management page is displayed.
2. Click **Create Job**. On the displayed job creation page, set parameters by referring to **Table 6-24** and click **OK**. The job development page is displayed.

Table 6-24 Parameters for creating a job

Parameter	Description	Example Value
Type	Job type, which can be **Flink SQL** or **Flink Jar**.	Flink SQL
Name	Job name, which can contain a maximum of 64 characters. Only letters, digits, and underscores (_) are allowed.	flinktest2
Job Type	Type of the job data source, which can be a stream job or a batch job.	Stream job
Description	Job description, which can contain a maximum of 100 characters.	-

Step 5 On the job development page, enter the following statements and click **Check Semantic** to check the input content.

```
CREATE TABLE kafka_source (
  id int,
  address string,
  proctime as PROCTIME()
) WITH (
  'connector' = 'kafka',
  'topic' = 'test_source',
  'properties.bootstrap.servers' = '<i>IP address of the Kafka broker instance</i>:<i>Kafka port</i>',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',--This parameter is not required for clusters in normal mode. Delete the comma (,) in the previous line.
  'properties.security.protocol' = 'SASL_PLAINTEXT',--This parameter is not required for clusters in normal mode.
  'properties.kerberos.domain.name' = 'hadoop.<i>System domain name</i>'--This parameter is not required for clusters in normal mode.
);
CREATE TABLE kafka_sink(
  id int,
  address string,
  name string
) WITH (
  'connector' = 'kafka',
  'topic' = 'test_sink',
  'properties.bootstrap.servers' = '<i>IP address of the Kafka broker instance</i>:<i>Kafka port</i>',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',--This parameter is not required for clusters in normal mode. Delete the comma (,) in the previous line.
  'properties.security.protocol' = 'SASL_PLAINTEXT',--This parameter is not required for clusters in normal mode.
  'properties.kerberos.domain.name' = 'hadoop.<i>System domain name</i>'--This parameter is not required for clusters in normal mode.
);
CREATE CATALOG myhive WITH (
  'type' = 'hive',
  'hive-version' = '3.1.0',
```



```
  'default-database' = 'default',
  'cluster.name' = 'flink_hive1'
);
use catalog myhive;
set
  table.sql-dialect = hive;
create table if not exists hive3 (id int, name string) stored as textfile TBLPROPERTIES (
  'streaming-source.enable' = 'false',
  'streaming-source.partition.include' = 'all',
  'lookup.join.cache.ttl' = '5 min'
);
INSERT INTO
  default_catalog.default_database.kafka_sink
select
  t1.id,
  t1.address,
  t2.name
from
  default_catalog.default_database.kafka_source as t1
  join hive3 FOR SYSTEM_TIME AS OF t1.proctime AS t2 ON t1.id = t2.id;
```

NOTE

- The IP address and port number of the Kafka broker instance are as follows:
  - To obtain the instance IP address, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka**, click **Instance**, and query the instance IP address on the instance list page.
  - If Kerberos authentication is enabled for the cluster (the cluster is in security mode), the Broker port number is the value of **sasl.port**. The default value is **21007**.
  - If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), the broker port number is the value of **port**. The default value is **9092**. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. Click **Configurations** then **All Configurations**. On the page that is displayed, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- The value of **'cluster.name'** is the name of the cluster connection created in **Step 3**.
- *System domain name*: You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.

Step 6 After the job is developed, in **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **60000**, and retain the default value for **Mode**.

Step 7 Click **Submit** in the upper left corner to submit the job.

Step 8 After the job is successfully executed, choose **More** > **Job Monitoring** to view the job running details.

Step 9 Execute the following commands to view the topic and write data to Kafka. For details, see **Managing Messages in Kafka Topics**.

./kafka-topics.sh --list ***--bootstrap-server*** *Service IP address of the Kafka Broker instance*:*Kafka port* **--command-config** *Client directory*/**Kafka/kafka/config/client.properties**

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka Broker instance is deployed*:*Kafka port* **--topic** *Topic name* **--producer.config** *Client directory*/**Kafka/kafka/config/producer.properties**

Enter the message content.

```
1,city1
2,city2
```


Press **Enter** to send the message.

Step 10 Run the following command to check whether data is received in the sink table, that is, check whether data is properly written to the Kafka topic after **Step 9** is performed. For details, see **Managing Messages in Kafka Topics**.

sh kafka-console-consumer.sh --topic test_sink --bootstrap-server *Service IP address of the Kafka broker instance*:*Kafka port number* --**consumer.config /opt/client/Kafka/kafka/config/consumer.properties**

The result is as follows:

```
1,city1,aname
2,city2,bname
```

----End

6.5.9 Creating a FlinkServer Job to Write Data to a Hudi Table

This section applies to MRS 3.1.2 or later clusters.

Scenario

This section describes how to interconnect FlinkServer with Hudi through Flink SQL jobs. When you use Flink SQL to read data from or write data to Hudi, the TINYINT, SMALLINT, and TIME types cannot be defined.

Table 6-25 lists the read and write operations supported by Flink on Hudi COW and MOR tables.

Table 6-25 Flink SQL read and write operations on Hudi tables

Flink SQL	COW Table	MOR Table
Batch write	Supported	Supported
Batch read	Supported	Supported
Stream write	Supported	Supported
Stream read	Supported	Supported

Prerequisites

- The HDFS, Yarn, Flink, and Hudi services have been installed in a cluster.
- The client that contains the Hudi service has been installed, for example, in the **/opt/client** directory.
- Flink 1.12.2 or later and Hudi 0.9.0 or later are required.
- You have created a user assigned with the **FlinkServer Admin Privilege** (for example, **flink_admin**) for accessing the Flink web UI by referring to **Creating a FlinkServer Role**. The user has been added to the **hadoop**, **hive**, and **kafkaadmin** user groups and granted the **Manager_administrator** role.

Creating a Job

Step 1 Log in to Manager as user **flink_admin** and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link on the right of **Flink WebUI** to access the Flink web UI.

Step 2 Create a Flink SQL job by referring to **Creating a FlinkServer Job**. On the job development page, configure the job as follows: Enter the SQL statement. After the SQL statement passes the verification, start the job. The following SQL examples are added as three jobs and run in sequence.

In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **60000**, and retain the default value for **Mode**.

NOTE

- CheckPoint should be enabled on the Flink web UI because data is written to a Hudi table only when a Flink SQL job triggers CheckPoint. Adjust the CheckPoint interval based on service requirements. You are advised to set the interval to a large number.
- If the CheckPoint interval is too short, job exceptions may occur due to untimely data updates. It is recommended that the CheckPoint interval be configured at the minute level.
- Asynchronous compaction is required when a Flink SQL job writes an MOR table. For details about the parameter for controlling the compaction interval, visit Hudi official website **https://hudi.apache.org/docs/configurations.html**.
- The following content is available in MRS 3.2.1 or later. By default, Hudi table write statements use Flink status index. To use the bucket index, add the following parameters to the statement:

```
'index.type'='BUCKET',
'hoodie.bucket.index.num.buckets'='Number of buckets in each partition of a Hudi table'
'hoodie.bucket.index.hash.field'='recordkey.field'
```

  - **hoodie.bucket.index.num.buckets**: Number of buckets in each partition of a Hudi table. Data in each partition is stored in each bucket in hash mode. This parameter cannot be modified after being set during table creation or data writing for the first time. Otherwise, an exception occurs during data update.
  - **hoodie.bucket.index.hash.field**: Field for calculating the hash value during bucketing. The field must be a subset of the primary key. The default value is the primary key of the Hudi table. If this parameter is left blank, the default value **recordkey.field** is used.
- The following content is available in MRS 3.2.1 or later. The same Hudi table can be written by the bucket indexes across the Flink and Spark engines.

1. Job 1: This Flink SQL job writes data to an MOR table in streams.

```
CREATE TABLE stream_mor(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream_mor',
'table.type' = 'MERGE_ON_READ',
'hoodie.datasource.write.recordkey.field' = 'uuid',
'write.precombine.field' = 'ts',
'write.tasks' = '4'
);

CREATE TABLE kafka(
uuid VARCHAR(20),
name VARCHAR(10),
```



```
age INT,
ts INT,
`p` VARCHAR(20)
) WITH (
'connector' = 'kafka',
'topic' = 'writehudi',
'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port number',
'properties.group.id' = 'testGroup1',
'scan.startup.mode' = 'latest-offset',
'format' = 'json',
'properties.sasl.kerberos.service.name' = 'kafka',--This parameter is not required for clusters in normal mode. Delete the comma (,) in the previous line.
'properties.security.protocol' = 'SASL_PLAINTEXT',--This parameter is not required for clusters in normal mode.
'properties.kerberos.domain.name' = 'hadoop.System domain name'--This parameter is not required for clusters in normal mode.
);

insert into
stream_mor
select
*
from
kafka;
```

2. Job 2: This Flink SQL job writes data to a COW table in streams.

```
CREATE TABLE stream_write_cow(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream_cow',
'hoodie.datasource.write.recordkey.field' = 'uuid',
'write.precombine.field' = 'ts',
'write.tasks' = '4'
);

CREATE TABLE kafka(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) WITH (
'connector' = 'kafka',
'topic' = 'writehudi',
'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port number',
'properties.group.id' = 'testGroup1',
'scan.startup.mode' = 'latest-offset',
'format' = 'json',
'properties.sasl.kerberos.service.name' = 'kafka',--This parameter is not required for clusters in normal mode. Delete the comma (,) in the previous line.
'properties.security.protocol' = 'SASL_PLAINTEXT',--This parameter is not required for clusters in normal mode.
'properties.kerberos.domain.name' = 'hadoop.System domain name'--This parameter is not required for clusters in normal mode.
);

insert into
stream_write_cow
select
*
from
kafka;
```

3. Job 3: This Flink SQL job reads MOR and COW tables in streams, merges data, and outputs the merged data to Kafka. Verify the SQL statement of job 3 and

start it after job 1 and job 2 are started and their status is running. Otherwise, an error message may be displayed during SQL verification, indicating that the Hudi table directory cannot be found.

```
CREATE TABLE stream_mor(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream_mor',
'table.type' = 'MERGE_ON_READ',
'hoodie.datasource.write.recordkey.field' = 'uuid',
'write.precombine.field' = 'ts',
'read.tasks' = '4',
'read.streaming.enabled' = 'true',
'read.streaming.check-interval' = '5',
'read.streaming.start-commit' = 'earliest'
);
CREATE TABLE stream_write_cow(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream_cow',
'hoodie.datasource.write.recordkey.field' = 'uuid',
'write.precombine.field' = 'ts',
'read.tasks' = '4',
'read.streaming.enabled' = 'true',
'read.streaming.check-interval' = '5',
'read.streaming.start-commit' = 'earliest'
);

CREATE TABLE kafka(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) WITH (
'connector' = 'kafka',
'topic' = 'readhudi',
'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port',
'properties.group.id' = 'testGroup1',
'scan.startup.mode' = 'latest-offset',
'format' = 'json',
'properties.sasl.kerberos.service.name' = 'kafka',--This parameter is not required for clusters in normal mode. Delete the comma (,) in the previous line.
'properties.security.protocol' = 'SASL_PLAINTEXT',--This parameter is not required for clusters in normal mode.
'properties.kerberos.domain.name' = 'hadoop.System domain name'--This parameter is not required for clusters in normal mode.
);

insert into
kafka
select
*
from
stream_mor union all select * from stream_write_cow;
```


NOTE

- Kafka port
  - Value of **sasl.port** when **Authentication Mode** of the cluster is **Security Mode**, **21007** by default.
  - Value of **port** when **Authentication Mode** of the cluster is **Normal Mode**, **9092** by default. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. On the page that is displayed, click the **Configurations** tab then the **All Configurations** sub-tab. On the displayed page, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- *System domain name*: You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.

Step 3 Execute the following script to write data to Kafka. For details, see **Managing Messages in Kafka Topics**.

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka instances are deployed*:*Kafka port* **--topic** *Topic name* **--producer.config** *Client directory*/**Kafka/kafka/config/producer.properties**

In this example, the topic name is **writehudi**.

sh kafka-console-producer.sh --broker-list *IP address of the node where the Kafka instance is deployed*:*Kafka port* **----topic writehudi --producer.config /opt/client/Kafka/kafka/config/producer.properties**

Enter the message content.

```
{"uuid": "1","name":"a01","age":10,"ts":10,"p":"1"}
{"uuid": "2","name":"a02","age":20,"ts":20,"p":"2"}
```

Press **Enter** to send the message.

Step 4 Consumes Kafka topic data and reads the result of reading the Hudi table from Flink streams.

sh kafka-console-consumer.sh --bootstrap-server *IP address of the node where the Kafka Broker instance is deployed*:*Kafka port* **--topic** *Topic name* **--consumer.config** *Client directory*/**Kafka/kafka/config/consumer.properties --from-beginning**

In this example, the topic name is **readhudi**.

sh kafka-console-consumer.sh --bootstrap-server *IP address of the node where the Kafka Broker instance is deployed*:*Kafka port* **--topic readhudi --consumer.config /opt/client/Kafka/kafka/config/consumer.properties --from-beginning**

The read result is as follows (the sequence is not fixed):

```
{"uuid": "1","name":"a01","age":10,"ts":10,"p":"1"}
{"uuid": "2","name":"a02","age":20,"ts":20,"p":"2"}
{"uuid": "1","name":"a01","age":10,"ts":10,"p":"1"}
{"uuid": "2","name":"a02","age":20,"ts":20,"p":"2"}
```

----End

Precautions for Using FlinkSQL Lookup Join Hudi

(This topic is available for MRS 3.5.0 and later.)

- The **lookup.join.cache.ttl** parameter is used to control the loading period of dimension table data. The default value is 60 minutes.
- Do not use a Hudi table with over 100,000 rows of records as a dimension table, as the data will be loaded to Flink TaskManager Heap memory.
- The new and updated data in the dimension table can be loaded for calculation only after the next loading period.

The following is a SQL example:

```
CREATE TABLE hudimor(
  uuid VARCHAR(20),
  name VARCHAR(10),
  age INT,
  ts INT,
  `p` VARCHAR(20),
  PRIMARY KEY (uuid) NOT ENFORCED
) PARTITIONED BY (`p`) WITH (
  'connector' = 'hudi',
  'path' = 'hdfs://hacluster/tmp/hudimor',
  'table.type' = 'MERGE_ON_READ',
  'hoodie.datasource.write.recordkey.field' = 'uuid',
  'write.precombine.field' = 'ts',
  'lookup.join.cache.ttl' = '60min'
);
CREATE TABLE datagen(uuid varchar(20), proctime as PROCTIME()) WITH (
  'connector' = 'datagen',
  'rows-per-second' = '1'
);
CREATE TABLE blackhole (
  uuid VARCHAR(20),
  name VARCHAR(10),
  age INT,
  ts INT,
  `p` VARCHAR(20)
) WITH ('connector' = 'blackhole');
insert into
  blackhole
select
  t1.uuid as uuid,
  t2.name as name,
  t2.age as age,
  t2.ts as ts,
  t2.p as p
FROM
  datagen AS t1
  left JOIN hudimor FOR SYSTEM_TIME AS OF t1.proctime AS t2 ON t1.uuid = t2.uuid;
```


WITH Parameters

Table 6-26 WITH parameters

Mode	Configuration Item	Mandatory	Default Value	Description
Read	read.tasks	No	4	Parallelism of the tasks for reading the Hudi table.
	read.streaming.enabled	No	false	Whether to enable stream read.
	read.streaming.start-commit	No	By default, the latest commit is the start position.	Start position (closed interval) of incremental stream and batch consumption in **yyyyMMddHHmmss** format.
	read.end-commit	No	By default, the latest commit is the end position.	End position (closed interval) of incremental stream and batch consumption in **yyyyMMddHHmmss** format.
Write	write.tasks	No	4	Parallelism of the tasks for reading data from the Hudi table.
	index.bootstrap.enabled	No	false	Whether to enable index loading. If it is enabled, the latest data in the stored table is loaded to the state at a time. If incremental data needs to be synchronized to full data and there are offline Hoodie tables available, you can enable the index loading function to write data in real time and ensure that data is unique.
	write.index_bootstrap.tasks	No	4	If indexes are loaded slowly when a job is started, you can choose a larger value for this parameter. After that, the efficiency is improved, but checkpoints are blocked in the bootstrap phase.

Mode	Configuration Item	Mandatory	Default Value	Description
	compaction.async.enabled	No	true	Whether to enable online compaction
	compaction.schedule.enabled	No	true	Whether to generate a compression plan periodically. You are advised to enable this function even if online compaction is disabled.
	compaction.tasks	No	10	Parallelism of the tasks for compacting data in the Hudi table.
	index.state.ttl	No	7D	Duration for storing indexes. The default value is 7 days. If the value is less than 0, indexes are stored permanently. Indexes are the core data structure for determining whether data is duplicate. For long-time updates, for example, updating data generated one month ago, you need to increase the value of this parameter.

Synchronizing Metadata from Flink On Hudi to Hive

After this feature is enabled, Flink automatically creates a Hudi table on Hive and adds partitions to the Hudi table when writing data to it. Then services such as SparkSQL and Hive can read data form the Hudi table.

The metadata can be synchronized with either of the following methods. The JDBC mode is used as an example in the following steps.

This step is required for MRS 3.2.0 or later.

- Synchronizing metadata to Hive in JDBC mode

```
CREATE TABLE stream_mor(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream_mor',
'table.type' = 'MERGE_ON_READ',
'hive_sync.enable' = 'true',
'hive_sync.table' = 'Name of the table to be synchronized to Hive',
'hive_sync.db' = 'Name of the database to be synchronized to Hive',
'hive_sync.metastore.uris' = 'Value of hive.metastore.uris in the hive-site.xml file on the Hive client',
'hive_sync.jdbc_url' = 'Value of CLIENT_HIVE_URI in the component_env file on the Hive client'
);
```


NOTICE

- **hive_sync.jdbc_url**: If the value of **CLIENT_HIVE_URI** in the Hive client file **component_env** contains ****, delete ****.
- To use the Hive style partitioning, add the following parameters:
  - 'hoodie.datasource.write.hive_style_partitioning' = 'true'
  - 'hive_sync.partition_extractor_class' = 'org.apache.hudi.hive.MultiPartKeysValueExtractor'
- Flink on Hudi synchronizes data to Hive. Hudi is case sensitive, while Hive is case insensitive. You are not advised to use uppercase letters in fields of Hudi tables. Otherwise, data may fail to be read or written.

- Synchronizing metadata to Hive in HMS mode

```
CREATE TABLE stream_mor(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream_mor',
'table.type' = 'MERGE_ON_READ',
'hive_sync.enable' = 'true',
'hive_sync.table' = 'Name of the table to be synchronized to Hive',
'hive_sync.db' = 'Name of the database to be synchronized to Hive',
'hive_sync.mode' = 'hms',
'hive_sync.metastore.uris' = 'Value of hive.metastore.uris in the hive-site.xml file on the Hive client',
'properties.hive.metastore.kerberos.principal' = 'Value of hive.metastore.kerberos.principal in the hive-site.xml file on the Hive client'
);
```

Methods with JDBC:

Step 1 Log in to Manager as user **flink_admin** and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link on the right of **Flink WebUI** to access the Flink web UI.

Step 2 Create a Flink SQL job by referring to **Creating a FlinkServer Job**. On the job development page, configure the job as follows: Enter the SQL statement. After the SQL statement passes the verification, start the job.

In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **60000**, and retain the default value for **Mode**.

```
CREATE TABLE stream_mor2(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream_mor2',
'table.type' = 'MERGE_ON_READ',
'hoodie.datasource.write.recordkey.field' = 'uuid',
'write.precombine.field' = 'ts',
'write.tasks' = '4',
'hive_sync.enable' = 'true',
'hive_sync.table' = 'Name of the table to be synchronized to Hive, for example, stream_mor2',
'hive_sync.db' = 'Name of the database to be synchronized to Hive, for example, default',
```



```
'hive_sync.metastore.uris' = 'Value of hive.metastore.uris in the hive-site.xml file on the Hive client',
'hive_sync.jdbc_url' = 'Value of CLIENT_HIVE_URI in the component_env file on the Hive client'
);
CREATE TABLE datagen (
uuid varchar(20), name varchar(10), age int, ts INT, p varchar(20)
) WITH (
'connector' = 'datagen',
'rows-per-second' = '1',
'fields.p.length' = '1'
);insert into stream_mor2 select * from datagen;
```

Step 3 Wait for the Flink job to run for a period of time and continuously write the random test data generated by datagen to the Hudi table. You can click **More** > **Job Monitoring** to go to the native UI of Flink and view the job status.

Step 4 Log in to the node where the client is deployed, load environment variables, run the beeline command to log in to the Hive client, and run SQL statements to check whether the Hudi Sink table is successfully created on Hive and whether data can be read from the table.

cd /opt/hadoopclient

source bigdata_env

beeline

desc formatted default.stream_mor2;

select * from default.stream_mor2 limit 5;

show partitions default.stream_mor2;

----End

6.5.10 Creating a FlinkServer Job to Write Data to a Kafka Message Queue

This section applies to MRS 3.1.2 or later clusters.

Scenario

This section describes the data definition language (DDL) of Kafka as a source or sink table, as well as the WITH parameters and example code for creating a table, and provides guidance on how to perform operations on the FlinkServer job management page.

If your Kafka cluster is in security mode, the following example SQL statements can be used.

Prerequisites

- The HDFS, Yarn, Kafka, and Flink services have been installed in a cluster.
- The client that contains the Kafka service has been installed, for example, in the **/opt/client** directory.
- You have created a user assigned with the **FlinkServer Admin Privilege** (for example, **flink_admin**) for accessing the Flink web UI by referring to **Creating a FlinkServer Role**.

Creating a Job

Step 1 Log in to Manager as user **flink_admin** and choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link on the right of **Flink WebUI** to access the Flink web UI.

Step 2 Create a Flink SQL job by referring to **Creating a FlinkServer Job**. On the job development page, configure the job parameters as follows and start the job.

In **Basic Parameter**, select **Enable CheckPoint**, set **Time Interval(ms)** to **60000**, and retain the default value for **Mode**.

```
CREATE TABLE KafkaSource (
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `age` INT
) WITH (
  'connector' = 'kafka',
  'topic' = 'test_source',
  'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port number',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',--This parameter is not required for clusters in normal mode. Delete the comma (,) in the previous line.
  'properties.security.protocol' = 'SASL_PLAINTEXT',--This parameter is not required for clusters in normal mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'--This parameter is not required for clusters in normal mode.
);
CREATE TABLE KafkaSink(
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `age` INT
) WITH (
  'connector' = 'kafka',
  'topic' = 'test_sink',
  'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port number',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',--This parameter is not required for clusters in normal mode. Delete the comma (,) in the previous line.
  'properties.security.protocol' = 'SASL_PLAINTEXT',--This parameter is not required for clusters in normal mode.
  'properties.kerberos.domain.name' = 'hadoop.System domain name'--This parameter is not required for clusters in normal mode.
);
Insert into
  KafkaSink
select
  *
from
  KafkaSource;
```


NOTE

- Kafka port
  - Value of **sasl.port** when **Authentication Mode** of the cluster is **Security Mode**, **21007** by default.
  - Value of **port** when **Authentication Mode** of the cluster is **Normal Mode**, **9092** by default. If the port number is set to **9092**, set **allow.everyone.if.no.acl.found** to **true**. The procedure is as follows:

    Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**. On the page that is displayed, click the **Configurations** tab then the **All Configurations** sub-tab. On the displayed page, search for **allow.everyone.if.no.acl.found**, set it to **true**, and click **Save**.
- *System domain name*: You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the value of **Local Domain**.
- You need to restart Flink jobs after expanding a Kafka Topic partition if you are using Flink 1.15.0 or an earlier version. Otherwise, new partitions may not be detected and consumption data may be missed. Alternatively, you can enable Kafka Topic partition detection in Flink.

  You can add the **scan.topic-partition-discovery.interval** parameter to the WITH property of the SQL Kafka source table and set the parameter to a dynamic refresh interval, for example, **5min**.

Step 3 On the job management page, check whether the job status is **Running**.

Step 4 Run the following command to check whether data is received in the sink table, that is, check whether data is properly written to the Kafka topic after **Step 5** is performed. For details, see **Managing Messages in Kafka Topics**.

sh kafka-console-consumer.sh --topic test_sink --bootstrap-server *Service IP address of the Kafka Broker instance*:*Kafka port* --**consumer.config /opt/client/Kafka/kafka/config/consumer.properties**

Step 5 View the topic and write data to the Kafka topic by referring to **Managing Messages in Kafka Topics**. After the data is written, view the execution result in the window in **Step 4**.

./kafka-topics.sh --list *--bootstrap-server* *Service IP address of the Kafka Broker instance*:*Kafka port* **--command-config** *Client directory*/**Kafka/kafka/config/client.properties**

sh kafka-console-producer.sh --broker-list *IP address of the node where Kafka instances reside*:*Kafka port number* **--topic** *Topic name* **--producer.config** *Client directory*/**Kafka/kafka/config/producer.properties**

For example, if the topic name is **test_source**, the script is **sh kafka-console-producer.sh --broker-list** *IP address of the node where Kafka instances are deployed*:*Kafka port* **--topic test_source** --**producer.config /opt/client/Kafka/kafka/config/producer.properties**.

Enter the message content.

```
1,clw,33
```

Press **Enter** to send the message.

----End

WITH Parameters

Parameter	Mandatory	Type	Description
connector	Yes	String	Connector to be used. **kafka** is used for Kafka.
topic	• Yes (Kafka functions as a sink table.) • No (Kafka functions as a source table.)	String	Topic name. • When the Kafka is used as a source table, this parameter indicates the name of the topic from which data is read. Topic list is supported. Topics are separated by semicolons (;), for example, **Topic-1; Topic-2**. • When Kafka is used as a sink table, this parameter indicates the name of the topic to which data is written. Topic list is not supported for sinks.
topic-pattern	No (Kafka functions as a source table.)	String	Topic pattern. This parameter is available when Kafka is used as a source table. The topic name must be a regular expression. **NOTE** **topic-pattern** and **topic** cannot be set at the same time.
properties.bootstrap.servers	Yes	String	List of Kafka brokers, which are separated by commas (,).
properties.group.id	Yes (Kafka functions as a source table.)	String	Kafka user group ID.
format	Yes	String	Format of the value used for deserializing and serializing Kafka messages.
properties.*	No	String	Authentication-related parameters that need to be added in security mode.
scan.topic-partition-discovery.interval	No	Duration	Interval at which the consumer dynamically discovers the created partition. Default value: **5 min**

6.6 Managing FlinkServer Jobs

6.6.1 Viewing the Health Status of FlinkServer Jobs

This topic is available for MRS 3.3.0 or later only.

Job Health Status

When a large number of Flink jobs are running in a cluster, the FlinkServer web UI provides the Flink job health management function to help you evaluate the health status of each job. You can view the health status of the current job on the page and export the health information of all jobs with just one click.

- **Healthy**: The job is healthy and running properly.
- **Subhealthy**:
  - The "ALM-45637 Flink Task Is Continuously Under Back Pressure" alarm is generated. After the alarm is cleared based on the alarm information, the health status automatically recovers to **Healthy**.
  - The "ALM-45639 Checkpointing of a Flink Job Times Out" alarm is generated. After the alarm is cleared based on the alarm information, the health status automatically recovers to **Healthy**.
- **Unhealthy**:
  - The "ALM-45636 Flink Job Checkpoints Keep Failing" alarm is generated. After the alarm is cleared based on the alarm information, the health status automatically recovers to **Healthy**.
  - The "ALM-45638 Number of Restarts After Flink Job Failures Exceeds the Threshold" alarm is generated. After the alarm is cleared based on the alarm information and the job is restarted. The health status automatically recovers to **Healthy**.

Prerequisites

- The cluster is running properly and the cluster client has been installed.
- The function of registering jobs with FlinkServer and the job alarming function have been enabled by configuring the *Client installation path*/**Flink/flink/conf/flink-conf.yaml** file before submitting a job. The parameter settings are as follows.

Table 6-27 Parameters for enabling job registration and job alarming

Parameter	Value	Description
job.register.enable	true	Whether to enable job registration with FlinkServer. Value options are as follows: • **true**: Enable • **false**: Disable

Parameter	Value	Description
job.alarm.enable	true	Whether to enable job alarming. Value options are as follows: • **true**: Enable • **false**: Disable

NOTE

If the function of registering jobs with FlinkServer is not enabled, you cannot start, develop, or stop jobs registered with FlinkServer through the client on the FlinkServer web UI.

- Ensure that the job is not submitted in session mode and the job name must be specified.

Viewing Job Health

Step 1 Access the Flink web UI. For details, see **Accessing the FlinkServer Web UI**.

Step 2 Click **Job Management**. The job management page is displayed.

- Viewing job health

  On the **Job Management** page, view the health status of the current job.

  – Empty: The job is not running and has no health status.
  – Green icon: healthy
  – Yellow icon: subhealthy
  – Red icon: unhealthy

- Exporting all job health reports

  Click **Job Health Report**. The system automatically exports the health status information of all jobs to the local host, including the job name, health status, submission user, alarm information, configuration information, and start time.

  – If the health score is **0**, the job is healthy.
  – If the health score is **1**, the job is subhealthy.
  – If the health score is **2**, the job is unhealthy.

----End

6.6.2 Importing and Exporting FlinkServer Job Information

This topic is available for MRS 3.2.0 or later only.

Importing and Exporting Jobs

The FlinkServer web UI enables you only to import and export jobs, UDFs, andstream tables.

- Jobs, flow tables, and UDFs with the same name cannot be imported to the same cluster.

- When exporting a job, you need to manually select the stream tables and UDFs on which the job depends. Otherwise, a dialog box indicating that the dependent data is not selected will be displayed. The application information of a job will not be exported.
- When you export a stream table, the application information on which the stream table depends will not be exported.
- When you export UDFs, the application information on which the UDFs depend and information about jobs used by UDFs will not be exported.
- Data import and export between different applications. are supported.

NOTICE

When you import or export FlinkSQL jobs, the **password** field in the jobs will be left blank to meet security requirements. Before you submit jobs, manually enter the password.

Importing a Job

Step 1 Access the Flink web UI as a user with **FlinkServer Admin Privilege**. For details, see **Accessing the FlinkServer Web UI**.

Step 2 Choose **System Management** > **Import Jobs**.

Step 3 Click **Select** to select a local TAR file and click **OK**. Wait until the file is imported.

NOTE

The maximum size of a local TAR file to be uploaded is 200 MB.

----End

Exporting a Job

Step 1 Access the Flink web UI as a user with **FlinkServer Admin Privilege**. For details, see **Accessing the FlinkServer Web UI**.

Step 2 Choose **System Management** > **Export Jobs**.

Step 3 Select the data to be exported in either of the following ways. To deselect the content, click **Clear Selected Node**.

- Select the data to be exported as required.
- Click **Query Regular Expression**. On the displayed page, select the type of the data to be exported (**Table Management**, **Job Management**, or **UDF Management**), enter the keyword, and click **Query**. After the data is successfully matched, click **Synchronize**.

  NOTE

  All matched data will be synchronized after you click **Synchronize**. Currently, you cannot select some data for synchronization.

Step 4 Click **Verify**. After the verification is complete, click **OK**. Wait until the data is exported.

----End

6.6.3 Configuring Automatic Clearing of FlinkServer Job Residuals

Scenario

If a Flink task stops unexpectedly, some directories may reside in the ZooKeeper and HDFS services. To delete the residual directories, set **ClearUpEnabled** to **true**.

Procedure

Step 1 Log in to Manager.

Step 2 Choose **Cluster** > **Services** > **Flink**. On the displayed page, click the **Configurations** tab and then **All Configurations**, search for parameter **ClearUpEnabled**, and set it to **true**. For details about related parameters, see **Table 6-28**.

Table 6-28 Parameters for deleting the residual directories

Parameter	Description	Default Value	Value Range
ClearUpEnabled	Specifies whether to delete the residual directories. Set this parameter to **true** if residual directories need to be deleted; set it to **false** otherwise.	true	true and false
ClearUpPeriod	Specifies the period for deleting residual directories, in minutes.	1440	1440~2147483647
TrashDirectoryRetentionPeriod	Specifies the period for retaining residual directories, in minutes.	10080	10080~2147483647

Step 3 Click **Save**.

NOTICE

- This function deletes only the residual directories in the **/flink_base** directory of the ZooKeeper service and the **/flink/recovery** directory of the HDFS service. User-defined directories will not be deleted.
- This function does not delete the **checkpoints** directory of the HDFS service. You need to manually delete it.

----End

6.6.4 Configuring the FlinkServer Job Restart Policy

FlinkServer Job Restart Policies

Flink supports different restart policies to control whether and how to restart a job when a fault occurs. If no restart policy is specified, the cluster uses the default restart policy. You can also specify a restart policy when submitting a job. For details about how to configure such a policy on the job development page of MRS 3.1.0 or later, see **Creating a FlinkServer Job**.

The restart policy can be specified by configuring the **restart-strategy** parameter in the Flink configuration file *Client installation directory*/**Flink/flink/conf/flink-conf.yaml** or can be dynamically specified in the application code. The configuration takes effect globally. Restart policies include **failure-rate** and the following two default policies:

- **No restart**: If CheckPoint is not enabled, this policy is used by default.
- **Fixed-delay**: If CheckPoint is enabled but no restart policy is configured, this policy is used by default.

No restart Policy

When a fault occurs, the job fails and does not attempt to restart.

Configure the parameter as follows:

```
restart-strategy: none
```

fixed-delay Policy

When a fault occurs, the job attempts to restart for a fixed number of times. If the number of attempts exceeds the times you specified, the job fails. The restart policy waits for a fixed period of time between two consecutive restart attempts.

In the following example, a job fails if the job attempts to restart for three times at an interval of 10 seconds. Configure the parameters as follows:

```
restart-strategy: fixed-delay
restart-strategy.fixed-delay.attempts: 3
restart-strategy.fixed-delay.delay: 10 s
```

failure-rate Policy

When a job fails, the job restarts directly. If the failure rate exceeds the value you configured, the job is considered as failed. The restart policy waits for a fixed period of time between two consecutive restart attempts.

In the following example, a job is considered as failed if the job attempts to restart for three times at an interval of 10 minutes. Configure the parameters as follows:

```
restart-strategy: failure-rate
restart-strategy.failure-rate.max-failures-per-interval: 3
restart-strategy.failure-rate.failure-rate-interval: 10 min
restart-strategy.failure-rate.delay: 10 s
```


Selecting a Restart Policy

- If you do not want to retry a failed job, select the **No restart** policy.
- To retry a failed job, select the **failure-rate** policy. If the fixed-delay policy is used, the number of job failures may reach the maximum number of retries due to hardware faults such as network and memory faults. As a result, the job fails.

  To prevent repeated restarts when the failure-rate policy is used, configure parameters as follows:

  ```
  restart-strategy: failure-rate
  restart-strategy.failure-rate.max-failures-per-interval: 3
  restart-strategy.failure-rate.failure-rate-interval: 10 min
  restart-strategy.failure-rate.delay: 10 s
  ```

Creating a FlinkServer Job

The statements of a SQL job submitted on Flink Server are saved to the DBServer. In MRS 3.5.0 and later versions, Flink Server encrypts SQL storage by default to protect information. When "FlinkSQL" is displayed in the command output on the FlinkServer web UI, the **password** field in the SQL statement is left blank. Before you submit a job, enter the password. For a custom connector, the password field name must contain the keyword **password** to prevent the password being displayed on the page.

NOTE

Disabling SQL encryption storage may cause password leak. You are advised to retain the default setting. If you still need to disable the function, perform the following operations:

1. (Optional) Back up jobs and then delete all jobs. For details about how to back up and import jobs, see **Importing and Exporting FlinkServer Job Information**.
2. Change the value of **ENABLE_DB_ENCRYPT** to **false**.

   Log in to the active and standby FlinkServer nodes, set **ENABLE_DB_ENCRYPT** in the **$BIGDATA_HOME//FusionInsight_Flink_x.x.x/x_x_FlinkServer/etc/flinkserver_service.properties** file to **false**, save the file, and exit.
3. Restart the affected FlinkServer instance.

   On FusionInsight Manager, choose **Cluster** > **Services** > **Flink** > **Instances**, select all FlinkServer instances, click **More**, and select **Restart Instance** to restart the instances.

Step 1 Access the Flink web UI. For details, see **Accessing the FlinkServer Web UI**.

Step 2 Click **Job Management**. The job management page is displayed.

Step 3 Click **Create Job**. Create a Flink SQL job or Flink Jar job, enter job information, and click **OK**. The job is created and the job development page is displayed.

Step 4 (Optional) To develop a job immediately, configure the job on the job development page.

The system allows you to add a lock to a job. The user who locks the job has all permissions of the job. Other users do not have the permissions to develop, start, or delete the locked job. However, they can forcibly acquire the lock to obtain all permissions. After this function is enabled, you can **Lock** and **Unlock** a job, or click **Acquire Lock** to obtain job permissions.

NOTE

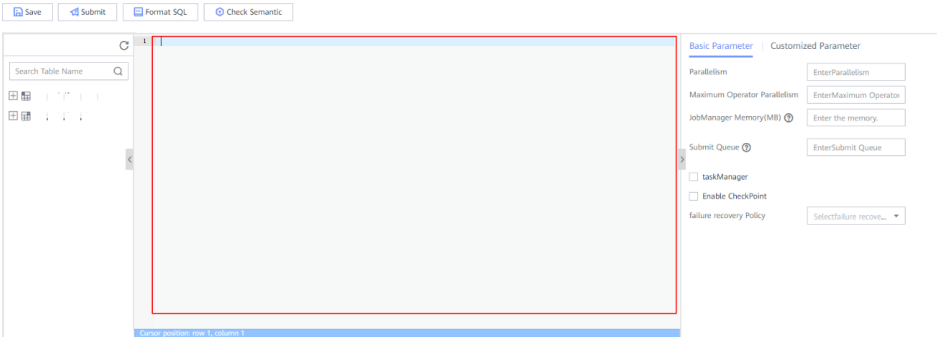
Job locks are enabled by default. You can view the status of this function on FusionInsight Manager. This topic is available for MRS 3.3.0 or later only.

Log in to FusionInsight Manager, choose **Cluster** > **Service** > **Flink**, click **Configuration** and then **All Configurations**, and search for the **job.edit.lock.enable** parameter. If the parameter value is **true**, the function is enabled. If the parameter value is **false**, the function is disabled.

- Creating a Flink SQL job

  a. Develop the job on the job development page.

  Figure 6-12 FlinkServer job development page

  b. Click **Check Semantic** to check the input content and click **Format SQL** to format SQL statements.

  c. Set basic and customized parameters as required by referring to **Table 6-29** and click **Save**.

  Table 6-29 Basic parameters

Parameter	Description
Parallelism	Number of parallel jobs
Maximum Operator Parallelism	Maximum degree of parallelism of operators
JobManager Memory (MB)	Memory of JobManager The minimum value is **4096**.
Submit Queue	Queue to which a job is submitted. If this parameter is not set, the **default** queue is used.
taskManager	taskManager running parameters. ▪ **Slots**: The default value is **1**. You are advised to set this parameter to the number of CPU cores. ▪ **Memory (MB)**: The minimum value is **4096**.

Parameter	Description
Enable CheckPoint	Whether to enable CheckPoint. After CheckPoint is enabled, you need to configure the following information: ■ **Time Interval (ms)**: This parameter is mandatory. ■ **Mode**: This parameter is mandatory. The options are **EXACTLY_ONCE** and **AT_LEAST_ONCE**. ■ **Minimum Interval (ms)**: The minimum value is **10**. ■ **Timeout Duration**: The minimum value is **10**. ■ **Maximum Parallelism**: The value must be a positive integer containing a maximum of 64 characters. ■ **Whether to clean up**: This parameter can be set to **Yes** or **No**. ■ **Whether to enable incremental checkpoints**: This parameter can be set to **Yes** or **No**.
Failure Recovery Policy	Failure recovery policy of a job. The options are as follows. For details, see **Configuring the FlinkServer Job Restart Policy**. ■ **fixed-delay**: You need to configure **Retry Times** and **Retry Interval (s)**. ■ **failure-rate**: You need to configure **Max Retry Times**, **Interval (min)**, and **Retry Interval (s)**. ■ **none**

d. Click **Submit** in the upper left corner to submit the job.

- Creating a Flink JAR job

  a. Click **Select** to upload a local JAR file and set parameters by referring to **Table 6-30** or add customized parameters.

Table 6-30 Parameter configuration

Parameter	Description
Local .jar File	Upload a local JAR file. Upload a local file smaller than the threshold specified by **flinkserver.upload.jar.max.size**. The default value is **500 MB**. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Flink** > **Configurations** > **All Configurations**, search for **flinkserver.upload.jar.max.size**, and set the JAR file threshold. The value ranges from **100 MB** to **5,120 MB**.
Main Class	Main-Class type. ▪ **Default**: By default, the class name is specified based on the **Mainfest** file in the JAR file. ▪ **Specify**: Manually specify the class name.
Class Name	Class name. This parameter is available when **Main Class** is set to **Specify**.
Class Parameter	Class parameters of Main-Class (parameters are separated by spaces).
Parallelism	Number of parallel jobs Concurrent tasks of each job operator. Appropriately increasing the value will improve the overall computing performance of a job. Considering switchover overheads due to increasing threads, the maximum value is four times the number of SPUs used by the computing unit. One to two times the number of SPUs of the computing unit is the optimal.
JobManager Memory (MB)	Memory of JobManager. The minimum value is **4096**.
Submit Queue	Queue to which a job is submitted. If this parameter is not set, the **default** queue is used.
taskManager	taskManager running parameters. ▪ **Slots**: The default value is **1**. You are advised to set this parameter to the number of CPU cores. ▪ **Memory (MB)**: The minimum value is **4096**.

b. Click **Save** to save the configuration and click **Submit** to submit the job.

Step 5 Return to the job management page. You can view information about the created job, including job name, type, status, kind, and description.

After a job is created, you can start, develop, stop, edit, and delete the job, view job details, and rectify checkpoint faults in the **Operation** column of the job.

NOTE

- To read files related to the submitted job on the node as another user, ensure that the user and the user who submitted the job belong to the same user group and the user has been assigned the FlinkServer application management role. For example, **application view** is selected by referring to **Creating a FlinkServer Role**.
- You can view details about jobs in the **Running** state.
- You can rectify checkpoint faults for jobs in the **Running failed**, **Running succeeded**, or **Stop** state.
- To set whether the checkpoints of failed or canceled jobs can be retained, log in to FusionInsight Manager and choose **Cluster** > **Services** > **Flink**, click **Configurations** and then **All Configurations**, search for and set the **execution.checkpointing.externalized-checkpoint-retention** parameter of FlinkServer.
  - **DELETE_ON_CANCELLATION**: Only checkpoints of failed jobs will be retained.
  - **RETAIN_ON_CANCELLATION** (default value in MRS3.5.0 or later): Checkpoints of failed or canceled jobs will be retained.
  - **NO_EXTERNALIZED_CHECKPOINTS**(default for MRS versions earlier than 3.5.0): Checkpoints of failed or canceled jobs will not be saved.

----End

6.6.5 Adding Third-Party Dependency JAR Packages to a FlinkServer Job

This topic is available for MRS 3.3.0 or later only.

Flink allows you to run user-defined Flink jobs using third-party dependency packages. You can upload and manage dependency JAR packages on the Flink web UI and invoke required dependencies when running jobs. The dependency management function does not support semantic verification. A dependency JAR package name must start with a letter, digit, or underscore (_) and cannot exceed 32 characters. The following third-party dependencies are supported:

- Custom connector dependency: After a custom connector JAR package is uploaded, **Dependency Type** is **connector** on the Flink web UI.
- Non-custom connector dependency: After a non-custom connector JAR package, such as a job dependency package, is uploaded, **Dependency Type** is **normal** on the Flink web UI.

To use this function, you need to prepare the dependency files. If you upload dependencies to the cluster by specifying a path, you need to create an HDFS path and upload the JAR package to the HDFS.

Uploading Dependency Packages

Step 1 Log in to FusionInsight Manager and access the Flink web UI. For details, see **Accessing the FlinkServer Web UI**.

Step 2 Click **Dependency Management**. The **Dependency Management** page is displayed.

Step 3 Click **Add Dependency**.

Table 6-31 Adding a dependency

Parameter	Description	Example
Customized connector or not	Whether the dependency is a custom connector. Set this parameter based on the site requirements. • **Yes**: The file is a custom connector dependency. • **No**: The file is a non-custom connector dependency.	Yes
Name	Name of the dependency, which must be the same as the connection name of connector in the uploaded dependency package. Dependency packages with the same name cannot be uploaded.	kafka
Register jar	Upload method of a JAR package: • **Upload File**: Upload a JAR package form the local host. • **Specify Path**: Upload a prepared dependency file from an HDFS path.	Uploading files
Upload File	If **Register jar** is set to **Upload File**, select a JAR file from the local host.	-
Specify Path	If **Register jar** is set to **Specify Path**, enter the HDFS path of the dependency file. (The JAR package must have been uploaded to the HDFS.)	/flink_upload_test/flink-connector-kafka-customization.jar
Description	Description of the dependency to be uploaded.	-

Step 4 Click **OK**

----End

Example

- Custom connector dependencies
  a. Upload a custom connector dependency by referring to **Uploading Dependency Packages**.
     For example, the dependency name is **kafka**, and the name of the custom connector JAR package is **flink-connector-kafka-customization.jar**.
  b. Create a SQL job by referring to **Creating a FlinkServer Job**. Set **connector** in the SQL statement to the dependency name, for example, **'connector'='kafka'**.


```
CREATE TABLE KafkaSinkTable (`user_id` INT, `name` VARCHAR) WITH (
  'connector' = 'kafka',
  'topic' = 'test_sink6',
  'properties.bootstrap.servers' = '192.168.20.134:21005',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'csv'
);
CREATE TABLE datagen (`user_id` INT, `name` VARCHAR) WITH (
  'connector' = 'datagen',
  'rows-per-second' = '5',
  'fields.user_id.kind' = 'sequence',
  'fields.user_id.start' = '1',
  'fields.user_id.end' = '1000'
);
insert INTO
  KafkaSinkTable
select
  *
from
  datagen;
```

- Non-Custom connector dependencies

  Upload a non-custom connector dependency for a job. For details, see **Uploading Dependency Packages**.

6.6.6 Using UDFs in FlinkServer Jobs

This section applies to MRS 3.1.2 or later clusters.

You can customize functions to extend SQL statements to meet personalized requirements. These functions are called user-defined functions (UDFs). You can upload and manage UDF JAR files on the Flink web UI and call UDFs when running jobs.

Flink supports the following three types of UDFs, as described in **Table 6-32**.

Table 6-32 Function classification

Type	Description
User-defined Scalar function (UDF)	Supports one or more input parameters and returns a single result value. For details, see **UDF Java and SQL Examples**.
User-defined aggregation function (UDAF)	Aggregates multiple records into one value. For details, see **UDAF Java and SQL Examples**.
User-defined table-valued function (UDTF)	Supports one or more input parameters and returns multiple rows or columns. For details, see **UDTF Java and SQL Examples**.

Uploading UDFs to FlinkServer

Step 1 Prepare a UDF JAR file that cannot exceed 200 MB.

Step 2 Access the Flink web UI. For details, see **Accessing the FlinkServer Web UI**.

Step 3 Click **UDF Management**. The **UDF Management** page is displayed.

Step 4 Click **Add UDF**. Select and upload the prepared UDF JAR file for **Local .jar File**.

Step 5 Enter the UDF name and description and click **OK**.

NOTE

- A maximum of 10 UDF names can be added. **Name** can be customized. **Class Name** must correspond to the UDF in the uploaded UDF JAR file.
- After the UDF JAR file is uploaded, the server retains the file for 5 minutes by default. If you click **OK** within 5 minutes, the UDF creation is complete. If you click **OK** after 5 minutes, the UDF creation fails and an error message is displayed, indicating that the local UDF file path is incorrect.

Step 6 In the UDF list, you can view information about all UDFs in the current application. You can edit or delete UDF information in the **Operation** column. (Only unused UDF items can be deleted.)

Step 7 (Optional) If you need to run or develop a job immediately, configure the job on the **Job Management** page. For details, see **Creating a FlinkServer Job**.

----End

UDF Java and SQL Examples

- UDF Java example

```
package com.xxx.udf;
import org.apache.flink.table.functions.ScalarFunction;
public class UdfClass_UDF extends ScalarFunction {
    public int eval(String s) {
        return s.length();
    }
}
```

- UDF SQL example

```
CREATE TEMPORARY FUNCTION udf as 'com.xxx.udf.UdfClass_UDF';
CREATE TABLE udfSource (a VARCHAR) WITH ('connector' = 'datagen','rows-per-second'='1');
CREATE TABLE udfSink (a VARCHAR,b int) WITH ('connector' = 'print');
INSERT INTO
  udfSink
SELECT
  a,
  udf(a)
FROM
  udfSource;
```

UDAF Java and SQL Examples

- UDAF Java example

```
package com.xxx.udf;
import org.apache.flink.table.functions.AggregateFunction;
public class UdfClass_UDAF {
    public static class AverageAccumulator  {
        public int sum;
    }
    public static class Average extends AggregateFunction<Integer, AverageAccumulator> {
        public void accumulate(AverageAccumulator acc, Integer value) {
            acc.sum += value;
        }
        @Override
        public Integer getValue(AverageAccumulator acc) {
            return acc.sum;
        }
        @Override
```



```
        public AverageAccumulator createAccumulator() {
            return new AverageAccumulator();
        }
    }
}
```

- UDAF SQL example

```
CREATE TEMPORARY FUNCTION udaf as 'com.xxx.udf.UdfClass_UDAF$Average';
CREATE TABLE udfSource (a int) WITH ('connector' = 'datagen','rows-per-
second'='1','fields.a.min'='1','fields.a.max'='3');
CREATE TABLE udfSink (b int,c int) WITH ('connector' = 'print');
INSERT INTO
  udfSink
SELECT
  a,
  udaf(a)
FROM
  udfSource group by a;
```

UDTF Java and SQL Examples

- UDTF Java example

```
package com.xxx.udf;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.table.functions.TableFunction;
public class UdfClass_UDTF extends TableFunction<Tuple2<String, Integer>> {
    public void eval(String str) {
        Tuple2<String, Integer> tuple2 = Tuple2.of(str, str.length());
        collect(tuple2);
    }
}
```

- UDTF SQL example

```
CREATE TEMPORARY FUNCTION udtf as 'com.xxx.udf.UdfClass_UDTF';
CREATE TABLE udfSource (a VARCHAR) WITH ('connector' = 'datagen','rows-per-second'='1');
CREATE TABLE udfSink (b VARCHAR,c int) WITH ('connector' = 'print');
INSERT INTO
  udfSink
SELECT
  str,
  strLength
FROM
  udfSource,lateral table(udtf(udfSource.a)) as T(str,strLength);
```

Introduction to Flink UDF Reuse

This topic is available for MRS 3.3.0 or later only.

The UDF reuse function is added to Flink SQL. When a UDF is executed for multiple times, only the first result is copied for the Nth (N > 1) execution. This ensures data consistency between multiple UDF executions and ensures that the UDF is executed only once, improving operator performance.

When configuring a Flink job, you can set **table.optimizer.function-reuse-enabled** to **true** on the Flink job development page of the Flink server web UI to enable the UDF reuse function. For details, see **Creating a FlinkServer Job**.

The following is a reuse example:

- UDF

```
class ItemExist extends ScalarFunction {
  val items: mutable.Set[String] = mutable.Set[String]()

  def eval(item: String): Boolean = {
    val exist = items.contains(item);
```



```
    if (!exist) {
      items.add(item)
    }
    exist
  }
}
```

- SQL statement

  SELECT * FROM (SELECT `a`, IfExist(b) as `exist`, `c` FROM Table1) WHERE exist IS FALSE;

- Result
  - Return value when the UDF reuse function is disabled:

    ```
    a,true,c
    ```

    Because IfExist is executed once in the WHERE condition and the result is **false**, the data has been stored in the cache. When IfExist is executed again in SELECT, **true** is returned.
  - Return value when the UDF reuse function is enabled:

    ```
    a,false,c
    ```

6.6.7 Configuring the FlinkServer UDF Sandbox

You can upload third-party JAR packages, such as UDFs and dependencies, on the Flink web UI based on job requirements, and invoke dependencies when verifying and running SQL jobs. To ensure that the uploaded JAR file is secure, the sandbox function is enabled for Flink by default. You can set the sandbox permission by setting the **flinkserver.security.policy** parameter by referring to **Configuring Permission of a Specified JAR Package** and disable the sandbox by setting the **security.manager.enabled** parameter by referring to **Disabling the FlinkServer Sandbox**.

Permissions

A permission consists of the type (mandatory), name, and allowed operation.

- Default permission
  - Property read permission: **permission java.util.PropertyPermission "*", "read"**
  - Socket permission, allowing **connect** and **resolve** to all ports: **permission java.net.SocketPermission "*", "connect,resolve"**
- Standard permission

Table 6-33 File permission

Type	Name	Allowed Operation	Example
java.io.FilePermission	• Name of a specified file • -: all files in the directory and subdirectories • *: all files in the directory	• read • write • delete • execute	• The following permission allows all files to be read, written, deleted, and executed: **permission java.io.FilePermission "<< ALL FILES>>", "read,write,delete,execute";** • The following permission allows the read of the user's home directory: **permission java.io.FilePermission "${user.home}/-", "read";**

Table 6-34 Socket permission

Type	Name	Allowed Operation	Examples
java.net.SocketPermission	• *Hostname*:*Port* • *: all addresses and ports	• accept • listen • connect • resolve	• The following permission allows all socket operations: **permission java.net.SocketPermission ":1-", "accept,listen,connect,resolve";** • The following permission allows the establishment of connections to and resolution of specific websites: **permission java.net.SocketPermission ".abc.com:1-", "connect,resolve";**

Table 6-35 Property permission

Type	Name	Allowed Operation	Examples
java.util.PropertyPermission	JVM property name to be accessed	• read • write	The following permission allows standard read of Java properties: **permission java.util.PropertyPermission "java.", "read";**

Table 6-36 Runtime permission

Type	Name
java.lang.RuntimePermission	• **accessDeclaredMembers**: allows code to use reflection to access private or protected members in other classes. • **createClassLoader**: allows code to create a class loader instance. • **createSecurityManager**: allows code to create a security manager instance, which will allow programmatic implementations to control the sandbox. This is a high-risk operation. With this permission, the UDF can modify or disable the **SecurityManager** of a service. • **exitVM**: allows the code to shut down the entire VM. • **getClassLoader**: allows code to access the class loader to obtain a specific class. • **setContextClassLoader**: allows code to set the context class loader for a thread. • **setFactory**: allows code to create a socket factory. • **setIO**: allows code to redirect **System.in** and **System.out** or **System.err** input and output streams. • **setSecurityManager**: allows code to set the security manager. • **stopThread**: allows code to invoke the **stop()** method of the thread class.

Table 6-37 Security permission

Type	Name
java.security.SecurityPermission	• **createAccessControlContext**: allows you to create a context environment for an access controller. • **getPolicy**: allows the search of classes that can implement sandbox policies. • **setPolicy**: allows you to set a class that can implement the sandbox policy. This is a high-risk operation. With this permission, the UDF can modify the policy of a service.

Table 6-38 Reflection permission

Type	Name
java.lang.reflect.ReflectPermission	**suppressAccessChecks**: allows reflection to be used to check private variables of any class.

Table 6-39 All permission

Type	Name
java.security.AllPermission	None (permission to perform any operation)

NOTE

If a third-party JAR dependency is used and the following error message is displayed, the sandbox permission is required:

```
Caused by: java.security.AccessControlException: access denied ("java.io.FilePermission" "xxxx" "read")
at java.security.AccessControlContext.checkPermission(AccessControlContext.java:472)
at java.security.AccessController.checkPermission(AccessController.java:886)
at java.lang.SecurityManager.checkPermission(SecurityManager.java:549)
at java.lang.SecurityManager.checkRead(SecurityManager.java:888)
at java.io.File.exists(File.java:825)  at com.xxx.ExpireUDF.(ExpireUDF.java:19)
```

Configuring Permission of a Specified JAR Package

Step 1 Log in to FusionInsight Manager and access the Flink web UI. For details, see **Accessing the FlinkServer Web UI**.

Step 2 Check the storage path of the JAR package.

- Record the UDF storage path.

  Click **UDF Management**. In the UDF list, view and record the storage path.

- Record the storage path of the third-party dependency.

  Click **Dependency Management**. In the dependency list, view and record the storage path.

Step 3 Configure permission for the specified dependency.

Return to FusionInsight Manager, choose **Cluster** > **Services** > **Flink** > **Configurations** > **All Configurations** > **FlinkServer (Role)** > **Customization**, set **flinkserver.security.policy** as follows, and save the settings:

- Name: Enter the storage path recorded in **Step 2**. If you need to add more paths, click the plus sign (+).
- Value: permission value, which ends with a semicolon (;). For example, **permission java.util.PropertyPermission "*", "read";permission java.net.SocketPermission "*", "connect,resolve"**. For details, see **Permissions**.

Step 4 Restart the FlinkServer instance.

Click **Instances**, select all FlinkServer instances, choose **More** > **Restart Instance**, and operate as prompted.

----End

Disabling the FlinkServer Sandbox

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **Flink** > **Configurations** > **All Configurations**.

Step 3 Search for the **security.manager.enabled** parameter and set its value to **false**.

Step 4 Click **Save**.

Step 5 Click **Instances**, select all FlinkServer instances, choose **More** > **Restart Instance**, and operate as prompted.

----End

6.7 Enterprise-Class Enhancements of Flink

6.7.1 Flink SQL Syntax Enhancement

This topic is available for MRS 3.3.0 or later only.

FlinkSQL DISTRIBUTEBY

The DISTRIBUTEBY feature is added to Flink SQL to partition data based on specified fields. A single or multiple fields are supported, solving the problem where only data needs to be partitioned. The following are some examples:

```
SELECT /*+ DISTRIBUTEBY('id') */ id, name FROM t1;
SELECT /*+ DISTRIBUTEBY('id', 'name') */ id, name FROM t1;
SELECT /*+ DISTRIBUTEBY('id1') */ id as id1, name FROM t1;
```


Processing Late Data in Flink SQL Window Functions

Window functions are added to Flink SQL to support late data processing. Currently, late data is supported in the TUMBLE, HOP, OVER, and CUMULATE window functions. The following is an example:

```
CREATE TABLE T1 (
 `int` INT,
 `double` DOUBLE,
 `float` FLOAT,
 `bigdec` DECIMAL(10, 2),
 `string` STRING,
 `name` STRING,
 `rowtime` TIMESTAMP(3),
 WATERMARK for `rowtime` AS `rowtime` - INTERVAL '1' SECOND
) WITH (
 'connector' = 'values',
);

-- The fields of the sink must be the same as the input data of the window, but the sequence can be
different.
CREATE TABLE LD_SINK(
 `float` FLOAT, `string` STRING, `name` STRING,  `rowtime` TIMESTAMP(3)
) WITH (
 'connector' = 'print',
);

SELECT  /*+ LATE_DATA_SINK('sink.name'='LD_SINK') */
  `name`,
  MIN(`float`),
  COUNT(DISTINCT `string`)
FROM TABLE(
  TUMBLE(TABLE T1, DESCRIPTOR(rowtime), INTERVAL '5' SECOND))
GROUP BY `name`, window_start, window_end
```

This feature also supports the output of the start time and end time of the current window when the window receives late data. The time can be output by adding **window.start.field** and **window.end.field** to the hint. The field type must be **timestamp**. The following is an example:

```
CREATE TABLE LD_SINK(
 `float` FLOAT, `string` STRING, `name` STRING,  `rowtime` TIMESTAMP(3), `windowStart` TIMESTAMP(3),
`windowEnd` TIMESTAMP(3)
) WITH (
 'connector' = 'print',
);

SELECT  /*+ LATE_DATA_SINK('sink.name'='LD_SINK', 'window.start.field'='windowStart',
'window.end.field'='windowEnd') */
  `name`,
  MIN(`float`),
  COUNT(DISTINCT `string`)
FROM TABLE(
  TUMBLE(TABLE T1, DESCRIPTOR(rowtime), INTERVAL '5' SECOND))
GROUP BY `name`, window_start, window_end
```

Setting Source Parallelism

This topic is available for MRS 3.3.0 or later only.

Flink SQL allows you to use the **source.parallelism** parameter to set the number of concurrent source operators to deal with data skew and back pressure and improve job performance.

This feature changes the Forward partition of source and downstream operators to the Rebalance partition. When the number of concurrent source operators is

different from the number of concurrent downstream operators (**parallelisms**) and data disorder is not allowed, enable the **DISTRIBUTEBY** feature together with this feature. For details, see **Flink SQL Syntax Enhancement**.

The following example sets the number of concurrent source operators to 2 and enables the **DISTRIBUTEBY** feature:

```
CREATE TABLE KafkaSource (
`user_id` VARCHAR,
`user_name` VARCHAR,
 `age` INT
 ) WITH (
 'connector' = 'kafka',
 'topic' = 'test_source',
 'properties.bootstrap.servers' = 'Service IP address of the Kafka broker instance:Kafka port',
 'properties.group.id' = 'testGroup',
 'scan.startup.mode' = 'latest-offset',
 'format' = 'csv',
 'properties.sasl.kerberos.service.name' = 'kafka',
 'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.System domain name',
 -- Set the number of concurrent source operators.
 'source.parallelism' = '2'
 );
CREATE TABLE KafkaSink(
  `user_id` VARCHAR,
  `user_name` VARCHAR,
 `age` INT
 ) WITH (
  'connector' = 'kafka',
  'topic' = 'test_sink',
  'properties.bootstrap.servers' = 'Service IP address of the Kafka broker instance:Kafka port',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',
   'properties.security.protocol' = 'SASL_PLAINTEXT',
   'properties.kerberos.domain.name' = 'hadoop.System domain name'
 );
--  Insert into KafkaSink select user_id, user_name, age from KafkaSource (DISTRIBUTEBY disabled)
-- Enable DISTRIBUTEBY.
Insert into KafkaSink select/*+ DISTRIBUTEBY('user_id') */ user_id, user_name, age from KafkaSource;
```

6.7.2 Table-Level TTL for Stream Joins

This topic is available for MRS 3.3.0 or later only.

When you join two Flink streams, there is a possibility that data in one table changes rapidly (short TTL) and data in the other table changes slowly (long TTL). Currently, Flink supports only table-level TTL. To ensure join accuracy, you need to set the table-level TTL to a long time. In this case, a large amount of expired data is stored in state backends, causing great workload pressure. To reduce the pressure, you can use Hints to set different expiration time for left and right tables. The WHERE clause is not supported.

For example, set the TTL of the left table (**state.ttl.left**) to 60 seconds and that of the right table (**state.ttl.right**) to 120 seconds.

- Use Hints in the following format:
  ```
  /*+ OPTIONS('state.ttl.left'='60S', 'state.ttl.right'='120S') */
  ```
- The following is a configuration example with a SQL statement:
  - Example 1:
    ```
    CREATE TABLE user_info (`user_id` VARCHAR, `user_name` VARCHAR) WITH (
     'connector' = 'kafka',
     'topic' = 'user_info_001',
    ```



```
  'properties.bootstrap.servers' = '192.168.64.138:21005',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv'
);
CREATE table print(
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `score` INT
) WITH ('connector' = 'print');
CREATE TABLE user_score (user_id VARCHAR, score INT) WITH (
  'connector' = 'kafka',
  'topic' = 'user_score_001',
  'properties.bootstrap.servers' = '192.168.64.138:21005',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv'
);
INSERT INTO
  print
SELECT
  t.user_id,
  t.user_name,
  d.score
FROM
  user_info as t
  JOIN
  --  Set different TTLs for left and right tables.
  /*+ OPTIONS('state.ttl.left'='60S', 'state.ttl.right'='120S') */
  user_score as d ON t.user_id = d.user_id;
```

- Example 2

```
INSERT INTO
  print
SELECT
  t1.user_id,
  t1.user_name,
  t3.score
FROM
  t1
  JOIN
  --  Set different TTLs for left and right tables.
  /*+ OPTIONS('state.ttl.left' = '60S', 'state.ttl.right' = '120S') */
  (
    select
      UPPER(t2.user_id) as user_id,
      t2.score
    from
      t2
  ) as t3 ON t1.user_id = t3.user_id;
```

6.7.3 Configuring Flink SQL Client to Support SQL Verification

This topic is available for MRS 3.3.0 or later only.

Configuration Method

You can verify SQL syntax during SQL job development on the SQL Client. Running SQL commands in verification mode does not start Flink jobs.

- Verifying SQL statements
  - When you run the SQL shell command, add **-v** or **--validate** to enter the verification mode.

    sql-client.sh -v

  - When you run the SQL shell command, you can run SET to enter or exit the verification mode.
    - Enter the verification mode: **SET table.validate = true;**
    - Exit the verification mode: **SET table.validate = false;**
- Verifying SQL scripts

  When you use the **-f** parameter to specify a SQL script, add **-v** to enter the verification mode.

  sql-client.sh -f *test.sql* **-v**

Submitting a Job Using the Flink SQL Client

Step 1 Install the cluster client, for example, in **/opt/hadoopclient**.

Step 2 Log in to the node where the client is installed as the client installation user.

Step 3 Run the following command to go to the client installation directory:

cd /opt/hadoopclient

Step 4 Run the following command to initialize environment variables:

source /opt/hadoopclient/bigdata_env

Step 5 Log in to the Flink SQL client and submit a job.

1. Start a yarn-session by referring to **Using the Flink Client** and record the yarn-session ID (**yid**).

   yarn-session.sh -nm "*session-name***"**
2. Run the following command to access the Flink SQL Client:

   cd /opt/hadoopclient/Flink/flink/bin

   ./sql-client.sh

Figure 6-13 Accessing the Flink SQL Client

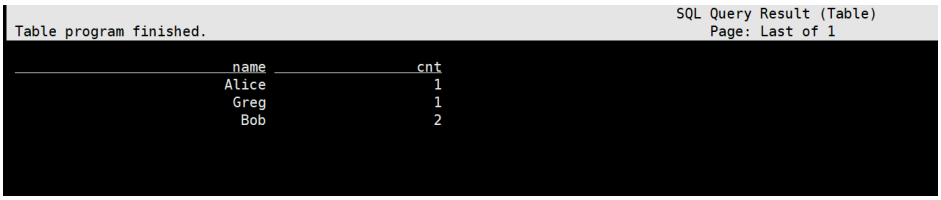
3. Set **high-availability.cluster-id** to the yarn-session ID.

   SET high-availability.cluster-id=*yarn-session ID***;**

4. Run the following SQL statement. If the execution is successful, the following information is displayed on the console.

   SELECT name, COUNT(*) AS cnt FROM (VALUES ('Bob'), ('Alice'), ('Greg'), ('Bob')) AS NameTable(name) GROUP BY name;

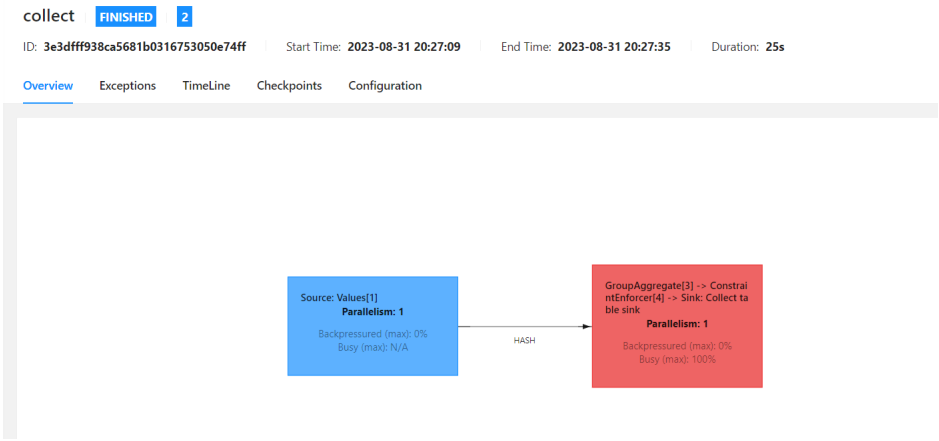
Figure 6-14 Execute result



5. View the executed job on Yarn.

   Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Yarn** > **Dashboard**, and click the link next to **ResourceManager WebUI** to go to the Yarn web UI and view the job.

Figure 6-15 Job



----End

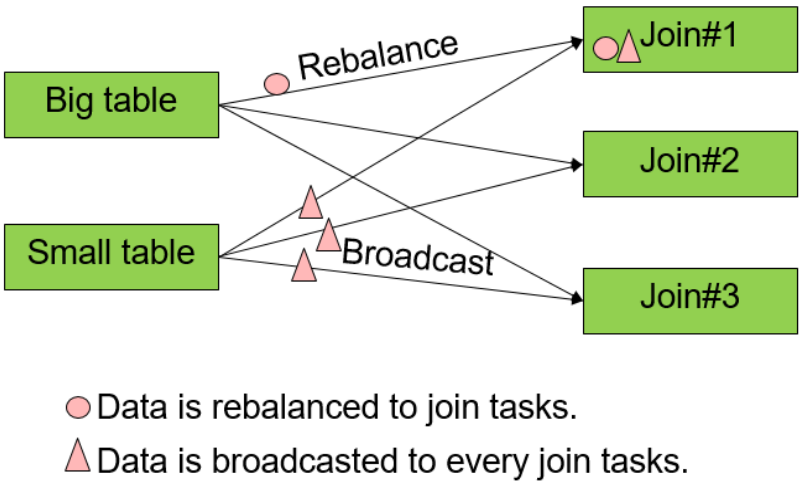
6.7.4 Enhancing the Joins of Large and Small Tables in Flink Jobs

This topic is available for MRS 3.3.0 or later only.

Joining Big and Small Tables

There are big tables and small tables when you join two Flink streams. Small table data is broadcasted to every join task, and large table data is rebalanced (distribute the data in a round robin fashion) to join tasks. This way, Flink SQL usability and job stability are improved.

Figure 6-16 Joining big and small tables



You can use Flink SQL Hints to specify the left or right table in a join as the broadcasted table and the other table as the rebalanced table. In the following SQL statement examples, table A and table C are small tables:

- Use table A as the broadcasted table.
  - Join
    ```
    SELECT /*+ BROADCAST(A) */ a2, b2 FROM A JOIN B ON a1 = b1
    ```
  - Where
    ```
    SELECT /*+ BROADCAST(A) */ a2, b2 FROM A, B WHERE a1 = b1
    ```
- Use table A and table C as broadcasted tables.
  ```
  SELECT /*+ BROADCAST(A, C) */ a2, b2, c2 FROM A JOIN B ON a1 = b1 JOIN C ON a1 = c1
  ```

NOTE

- This feature can be used with **/*+ BROADCAST(smallTable1, smallTable2) */** to be compatible with the open-source joins of two streams.
- Switching between open-source joins and this feature is not supported because this feature broadcasts data to each join task.
- Using a small table as the left table of a LEFTJOIN is not supported. Using a small table as the right table of a RIGHTJOIN is not supported.

Deduplicating Data When Joining Big and Small Tables

When you join two streams, there is a possibility that the join operator receives a large amount of duplicate data sent by one stream. Downstream operators need to process a large amount of duplicate data, affecting job performance.

For example, join fields (P1, A1, and A2) in table A with fields (P1, B1, B2, and B3) in table B to generate field C. A large amount of data in table B is updated but the fields are remain unchanged. Assume that only the B1 and B2 fields are used in the join and only the B3 field is updated. When you update table B, B1 and B2 fields should be ignored with the deduplication function.

```
select  A.A1,B.B1,B.B2 from A join B on A.P1=B.P1
```

To deduplicate table B updates, you can use Hints to set deduplication for the left table (duplicate.left) or right table (duplicate.right).

- Format
  - Set deduplication for the left table.
    ```
    /*+ OPTIONS('duplicate.left'='true')*/
    ```
  - Set deduplication for the right table.
    ```
    /*+ OPTIONS('duplicate.right'='true')*/
    ```
  - Set deduplication for both tables.
    ```
    /*+ OPTIONS('duplicate.left'='true','duplicate.right'='true')*/
    ```
- The following is an example in a SQL statement:

  For example, set deduplication for both the left table **user_info** and the right table **user_score**.

  ```
  CREATE TABLE user_info (`user_id` VARCHAR, `user_name` VARCHAR) WITH (
    'connector' = 'kafka',
    'topic' = 'user_info_001',
    'properties.bootstrap.servers' = '192.168.64.138:21005',
    'properties.group.id' = 'testGroup',
    'scan.startup.mode' = 'latest-offset',
    'value.format' = 'csv'
  );
  CREATE table print(
    `user_id` VARCHAR,
    `user_name` VARCHAR,
    `score` INT
  ) WITH ('connector' = 'print');
  CREATE TABLE user_score (user_id VARCHAR, score INT) WITH (
  ```



```
  'connector' = 'kafka',
  'topic' = 'user_score_001',
  'properties.bootstrap.servers' = '192.168.64.138:21005',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv'
);
INSERT INTO
  print
SELECT
  t.user_id,
  t.user_name,
  d.score
FROM
  user_info as t
  JOIN
  -- Set deduplication for both left and right tables.
  user_score /*+ OPTIONS('duplicate.left'='true','duplicate.right'='true')*/ as d ON t.user_id =
d.user_id;
```

6.7.5 Exiting FlinkSQL OVER Window Upon Expiration

This topic is available for MRS 3.5.0 or later only.

The function of exiting the Flink SQL OVER window upon data expiration is added. When the existing data expires and no new data arrives, OVER aggregation results are updated and the latest calculation results are sent to the downstream operator. You can use this function by configuring the parameters as follows:

Table 6-40 Parameters for enabling the function of exiting the Flink SQL OVER window upon data expiration

Parameter	Default Value	Description
over.window.interval	-1	The interval between two pieces of adjacent data, in milliseconds. If the interval is exceeded, the OVER window registers the expiration trigger. Value options are as follows: ● **-1**: This function is disabled. ● **0**: An expiration trigger is registered for each piece of data, but the pressure on the job is increased. You can increase the value of **over.window.interval** based on the job requirements. ● Positive number: The interval between two pieces of adjacent data. If the interval is exceeded, the OVER window registers the expiration trigger.

Parameter	Default Value	Description
over.window.interval.last.rowdata	false	Whether to send only the last piece of data to the downstream operator. Value options are as follows: • **false** (default): All data that has not expired is sent. • **true**: Only the last piece of data is sent.

How to Use

When configuring a Flink job, you can add the custom parameter **over.window.interval** on the job development page of the FlinkServer web UI and set its value greater than or equal to 0 to enable the window to support the data expiration function. For details about how to create a job, see **Creating a FlinkServer Job**. This setting takes effect for all OVER windows in a job. You are advised to use this function for jobs with a single OVER window.

- Example SQL statement

```
CREATE TABLE OverSource (
  `rowtime` TIMESTAMP_LTZ(3),
  `groupId` INT,
  `value` INT,
  `name` STRING,
  `additional_field` STRING,
  `proctime` as PROCTIME(),
  WATERMARK FOR rowtime AS rowtime
) WITH (
 'connector' = 'kafka',
 'topic' = 'test_source',
 'properties.bootstrap.servers' = 'Service IP address of the Kafka broker instance:Kafka port',
 'properties.group.id' = 'testGroup',
 'scan.startup.mode' = 'latest-offset',
 'format' = 'csv',
 'properties.sasl.kerberos.service.name' = 'kafka',  --Delete this parameter if the cluster where FlinkServer is deployed is in normal mode.
 'properties.security.protocol' = 'SASL_PLAINTEXT',  --Delete this parameter if the cluster where FlinkServer is deployed is in normal mode.
 'properties.kerberos.domain.name' = 'hadoop.System domain name'  --Delete this parameter if the cluster where FlinkServer is deployed is in normal mode.
);

CREATE TABLE LD_SINK(
 `name` STRING, `groupId` INT, `rowtime` TIMESTAMP_LTZ(3),`count_zw` BIGINT,`sum_zw` BIGINT
) WITH (
 'connector' = 'print'
);

SELECT
  `name`,
  `groupId`,
  COUNT(`value`) OVER (
    PARTITION BY groupId
    ORDER BY
      proctime RANGE BETWEEN INTERVAL '10' second PRECEDING
      AND CURRENT ROW
  ) as count_zw,
  SUM(`value`) OVER (
    PARTITION BY groupId
    ORDER BY
```



```
    proctime RANGE BETWEEN INTERVAL '10' second PRECEDING
    AND CURRENT ROW
  ) as sum_zw
FROM
  OverSource
```

- SQL statement output

  For example, with **proctime** is **10:00:00**, **groupid** is **1**, and **value** is 1 to 5, five consecutive data records are imported to Flink.

  ```
  ["zw", 1, 1, 1]
  ["zw", 1, 2, 3]
  ["zw", 1, 3, 6]
  ["zw", 1, 4, 10]
  ["zw", 1, 5, 15]
  ```

  After 10:00:10, no data is imported and the window starts to exit.

  ```
  ["zw", 1, 4, 14]
  ["zw", 1, 3, 12]
  ["zw", 1, 2, 9]
  ["zw", 1, 1, 5]
  ["zw", 1, 0, 0]
  ```

6.7.6 Limiting Read Rate for Flink SQL Kafka and Upsert-Kafka Connector

This section applies to MRS 3.3.0 or later.

Scenarios

Traffic limiting is required when Kafka and upsert-kafka connector consume data.

How to Use

Add the **subtask.scan.records-per-second.limit** parameter to the created source stream table. This parameter indicates the number of Kafka records consumed in a single partition per second. The overall traffic on the source end is **min(source parallelism * subtask.scan.records-per-second.limit, kafka partitions num * subtask.scan.records-per-second.limit)**.

The following is a SQL example:

```
CREATE TABLE KafkaSource (
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `age` INT
) WITH (
  'connector' = 'kafka',
  'topic' = 'test_source',
  'properties.bootstrap.servers' = '<i>IP address of the Kafka broker instance</i>:<i>Kafka port</i>',
  'properties.group.id' = '<i>testGroup</i>',
  'scan.startup.mode' = 'latest-offset',
  'format' = 'csv',
  <b>'subtask.scan.records-per-second.limit' = '1000',</b>
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.<i>System domain name</i>'
);
CREATE TABLE KafkaSink(
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `age` INT
) WITH (
```



```
  'connector' = 'kafka',
  'topic' = 'test_sink',
  'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port',
  'scan.startup.mode' = 'latest-offset',
  'format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.System domain name'
);
Insert into
  KafkaSink
select
  *
from
  KafkaSource;
```

6.7.7 Consuming Data in drs-json Format with FlinkSQL Kafka Connector

This section applies to MRS 3.3.0 or later.

Scenarios

FlinkSQL needs to consume data in drs-json format (a CDC message format) in Kafka.

How to Use

In the created Kafka Connector Source stream table, set **format** to **drs-json**.

The following is a SQL example:

```
CREATE TABLE KafkaSource (
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `age` INT
) WITH (
  'connector' = 'kafka',
  'topic' = 'test_source',
  'properties.bootstrap.servers' = 'IP address of the Kafka broker instance:Kafka port',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'format' = 'drs-json',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.System domain name'
);
CREATE TABLE printSink(
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `age` INT
) WITH (
  'connector' = 'print'
);
Insert into
  printSink
select
  *
from
  KafkaSource;
```

6.7.8 Using ignoreDelete in JDBC Data Writes

This section applies to MRS 3.3.0 or later.

Scenarios

Data in DELETE and UPDATE_BEFORE states can be filtered out when FlinkSQL writes JDBC data.

How to Use

Add **filter.record.enabled** and **filter.row-kinds** parameters to a created JDBC Connector Sink stream table.

NOTE

- The default value of **filter.record.enabled** is **false**.
- The default value of **filter.row-kinds** is **UPDATE_BEFORE, DELETE**.

The following is a SQL example:

```
CREATE TABLE user_score (
  idx varchar(20),
  user_id varchar(20),
  score bigint
) WITH (
  'connector' = 'kafka',
  'topic' = 'topic-qk',
  'properties.bootstrap.servers' = 'xxxx:21005',
  'properties.group.id' = 'test_qk',
  'scan.startup.mode' = 'latest-offset',
  'format' = 'csv'
);
CREATE TABLE mysql_output (
  idx varchar(20),
  user_id varchar(20),
  all_score bigint,
  PRIMARY KEY(idx, user_id) NOT ENFORCED
) WITH(
  'connector' = 'jdbc',
  'url' = 'jdbc:mysql://IP address of the MySQL server.MySQL server port/mysql',
  'table-name' = 'customer_t1',
  'username' = 'username', --Username for connecting the MySQL database
  'password' = 'password',--Password for connecting the MySQL database
  'filter.record.enabled' = 'true',
  'filter.row-kinds' = 'UPDATE_BEFORE'
);
insert into
  mysql_output
select
  idx,
  user_id,
  sum(score) as all_score
from
  user_score
group by
  idx,
  user_id;
```

6.7.9 Join-To-Live

Flink dual-stream join needs to store data in the state backend. Currently, RocksDB is widely used as the state backend. In scenarios where the time to live (TTL) is too long, the TTL cannot be determined, or the data traffic increases, heavy traffic increases the state data and storage pressure. As a result, job stability decreases, or TTL expiration may cause inaccurate joins.

For services whose number of joins are determined, the Join-To-Live (JTL) feature can be used to reduce the pressure on state backends. This feature determines

whether data expires based on the number of joins. It can be configured in either of the following ways:

NOTE

- This function is available for the inner join statement of Flink regular joins only.
- This function cannot be used together with job-level TTLs, table-level TTLs, or small table broadcasting.
- A primary key must be specified for a table that has a specified JTL. Otherwise, the query result may be inaccurate.

- Method 1: Using through SQL hints

  eliminate-state.left.threshold: indicates the threshold of the number of associations on the left. If the number of associations on the left exceeds the threshold, the piece of data expires.

  eliminate-state.right.threshold: indicates the threshold of the number of associations on the right. If the number of associations on the right exceeds the threshold, the piece of data expires.

  Example 1:

  ```
  SELECT * FROM t1
    JOIN /*+ OPTIONS('eliminate-state.right.threshold'='1', 'eliminate-state.left.threshold'='2') */
    t2 ON a1 = a2
  ```

  Example 2:

  ```
  SELECT a1, a2, a3 from
    t1
    join /*+ OPTIONS('eliminate-state.left.threshold'='1', 'eliminate-state.right.threshold'='2') */
    t2
    on a1 = a2
    join /*+ OPTIONS('eliminate-state.left.threshold'='3', 'eliminate-state.right.threshold'='4') */
    t3
    on a2 = a3
  ```

- Method 2: Configuring the two parameters in *Client installation path*/**Flink/flink/conf/flink-conf.yaml** for globally effective

  ```
  table.exec.join.eliminate-state.left.threshold
  table.exec.join.eliminate-state.right.threshold
  ```

6.7.10 Row Filter

This section applies to MRS 3.3.1 or later.

Scenarios

When using Flink SQL, you can set search criteria of rows to authorize a user to access specified rows and hide other rows from them.

Prerequisites

- Kerberos authentication is enabled for the cluster (the cluster is in security mode). Ranger, Hive, Flink have been installed and are running properly.
- The user, user group, or role with required permissions have been created, and the user has been added to the **hive** group.
- This feature can be used only on the FlinkServer platform.
- The default dialect is required for Flink SQL.
- The case of the filter criteria fields must be the same as that of the Flink SQL fields.

- This feature is not available for Hive view tables.

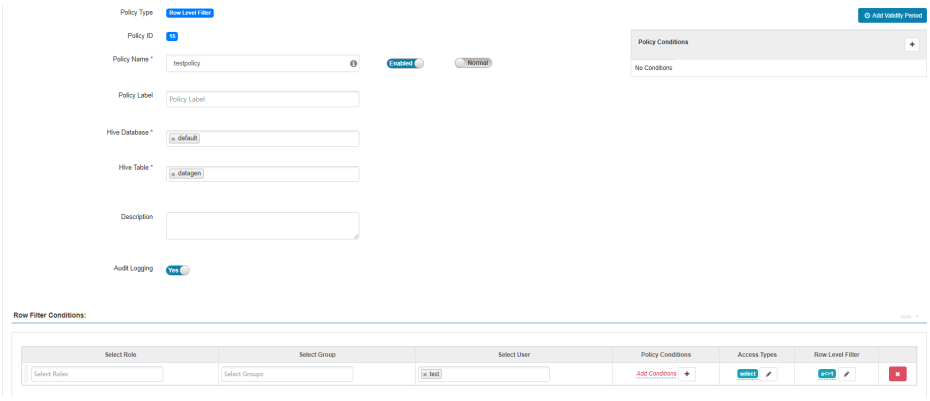
Configuring Flink SQL Row Filtering

Step 1 Configuring custom FlinkServer parameters.

1. Log in to FusionInsight Manager.
2. Choose **Cluster** > **Services** > **Flink** > **Configurations** > **All Configurations** > **FlinkServer(Role)** > **Customization** , and locate **flink.customized.configs**.
3. Add custom parameter **table.ranger.security.authorization**, set it to **true**, click **Save**, and save the settings.
4. Click the **Instances** tab, select all FlinkServer instances, choose **More** > **Restart Instance**.

Step 2 Configure Hive row filter for a specified user. For details, see **Hive Row-Level Data Filtering**.

For example, add the search criterion **a<>1** to the **datagen** table in the **default** database for the **test** user.

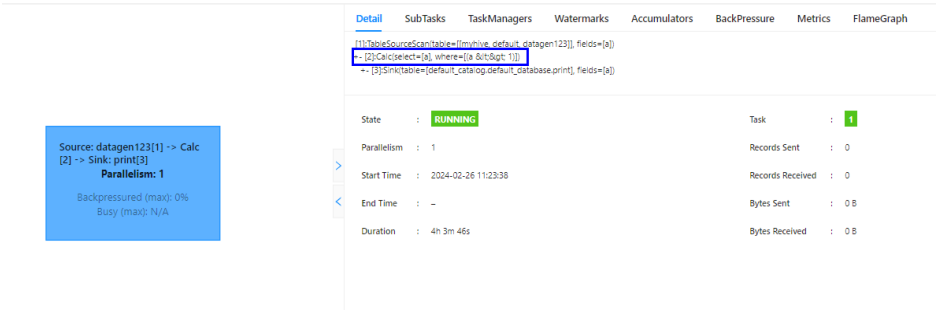


Step 3 Submit a Flink SQL job through FlinkServer as a user with the required permissions. For details, see **Creating a FlinkServer Job**.

The following is a SQL example:

```
CREATE CATALOG myhive WITH (
  'type' = 'hive',
  'hive-version' = '3.1.0',
  'default-database' = 'default',
  'cluster.name' = 'flink_hive'
);
CREATE TABLE print (a INT) WITH ('connector' = 'print');
CREATE TABLE IF NOT EXISTS myhive.`default`.datagen (a INT) WITH (
  'connector' = 'datagen',
  'rows-per-second' = '1'
);
set table.sql-dialect = default;
insert into
  print
select
  *
from
  myhive.`default`.datagen;
```

You can view the configured search criteria on the native Flink job page. In the following example, **a<>1** has been added to the criteria.



----End

6.7.11 FlinkSQL Operator Parallelism

This section applies to MRS 3.5.0 or later.

Scenarios

For jobs submitted through CompiledPlan, the parallelism and TTL of operators are subject to the values in CompiledPlan instead of the values in **flink-conf.yaml**. FlinkSQL allows you to set the parallelism of operators by modifying CompiledPlan of a job.

When you modify CompiledPlan, do not change the JSON file structure, or the job will fail to be submitted. The save path of CompiledPlan can be an HDFS path or an OBS path. In this example, the HDFS path is used.

How to Use

Change the value of **table.exec.resource.default-parallelism** of the an operator in CompiledPlan.

Example

Step 1 Develop a FlinkServer SQL job.

On the SQL development page of FlinkServer, enter the following SQL statements by referring to **Creating a FlinkServer Job** and click **Check Semantic**:

```
set parallelism.default = 2;

CREATE TABLE print(
  uuid VARCHAR(20),
  name VARCHAR(10),
  age INT,
  ts INT
) WITH (
  'connector' = 'print'
);
CREATE TABLE datagen(
  uuid VARCHAR(20),
  name VARCHAR(10),
  age INT,
  ts INT
) WITH (
  'connector' = 'datagen',
  'rows-per-second' = '1'
);
COMPILE PLAN 'hdfs://hacluster/tmp/plan.json' FOR insert into
  print
```



```
select
  *
from
  datagen;
```

Step 2 View and modify the operator parallelism in the CompiledPlan file.

1. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS**, click the link next to **NameNode Web UI**, and choose **Utilities** > **Browse the file system** to view and obtain the **hdfs://hacluster/tmp/plan.json** file. Click **Check Semantic**. The CompiledPlan file is generated in the HDFS. The file content is similar to the following example. The parallelism of the datagen operator (ID: 1) is **2**.
2. For example, to change the parallelism of the datagen operator to **1**, modify **"table.exec.resource.default-parallelism": "1"** and save, and then upload it to **hdfs://hacluster/tmp/plan.json**.

```
{
  "flinkVersion" : "1.17",
  "nodes" : [ {
    "id" : 1,
    "type" : "stream-exec-table-source-scan_1",
    "configuration" : {
      "table.exec.resource.default-parallelism" : "2"
    },
    "scanTableSource" : {
      "table" : {
        "identifier" : "`default_catalog`.`default_database`.`datagen`",
        "resolvedTable" : {
          "schema" : {
            "columns" : [ {
              "name" : "uuid",
              "dataType" : "VARCHAR(20)"
            }, {
              "name" : "name",
              "dataType" : "VARCHAR(10)"
            }, {
              "name" : "age",
              "dataType" : "INT"
            }, {
              "name" : "ts",
              "dataType" : "INT"
            } ],
            "watermarkSpecs" : [ ]
          },
          "partitionKeys" : [ ],
          "options" : {
            "connector" : "datagen",
            "rows-per-second" : "1"
          }
        }
      }
    },
    "outputType" : "ROW<`uuid` VARCHAR(20), `name` VARCHAR(10), `age` INT, `ts` INT>",
    "description" : "TableSourceScan(table=[[default_catalog, default_database, datagen]], fields=[uuid,
name, age, ts])",
    "inputProperties" : [ ]
  }, {
    "id" : 2,
    "type" : "stream-exec-sink_1",
    "configuration" : {
      "table.exec.resource.default-parallelism" : "2",
      "table.exec.sink.keyed-shuffle" : "AUTO",
      "table.exec.sink.not-null-enforcer" : "ERROR",
      "table.exec.sink.rowtime-inserter" : "ENABLED",
      "table.exec.sink.type-length-enforcer" : "IGNORE",
      "table.exec.sink.upsert-materialize" : "AUTO"
```



```
  },
  "dynamicTableSink" : {
    "table" : {
      "identifier" : "`default_catalog`.`default_database`.`print`",
      "resolvedTable" : {
        "schema" : {
          "columns" : [ {
            "name" : "uuid",
            "dataType" : "VARCHAR(20)"
          }, {
            "name" : "name",
            "dataType" : "VARCHAR(10)"
          }, {
            "name" : "age",
            "dataType" : "INT"
          }, {
            "name" : "ts",
            "dataType" : "INT"
          } ],
          "watermarkSpecs" : [ ]
        },
        "partitionKeys" : [ ],
        "options" : {
          "connector" : "print"
        }
      }
    }
  },
  "inputChangelogMode" : [ "INSERT" ],
  "inputProperties" : [ {
    "requiredDistribution" : {
      "type" : "UNKNOWN"
    },
    "damBehavior" : "PIPELINED",
    "priority" : 0
  } ],
  "outputType" : "ROW<`uuid` VARCHAR(20), `name` VARCHAR(10), `age` INT, `ts` INT>",
  "description" : "Sink(table=[default_catalog.default_database.print], fields=[uuid, name, age, ts])"
 } ],
 "edges" : [ {
  "source" : 1,
  "target" : 2,
  "shuffle" : {
    "type" : "FORWARD"
  },
  "shuffleMode" : "PIPELINED"
 } ]
}
```

Step 3 Submit the FlinkServer SQL job again.

Return to the FlinkServer SQL development page, enter the following SQL statement, and submit:

```
EXECUTE PLAN 'hdfs://hacluster/tmp/plan.json';
```

----End

6.7.12 Optimizing FlinkSQL JSON_VALUE Performance

This section applies to MRS 3.5.0 or later.

Scenarios

When the built-in JSON_VALUE function parses multiple fields of a JSON item, the parsing result of the previous JSON item is reused to improve the performance.

How to Use

When you configure a Flink job, set **table.optimizing.json-value-optimization** to **true** on the Flink job development page of the FlinkServer web UI to optimize the performance of the JSON_VALUE function. For details, see **Creating a FlinkServer Job**.

The following is an example SQL statement:

```
create table kafkaSource (
    msgValue STRING
) with (
    'connector' = 'kafka'
    ,'topic' = 'topic-1'
    ,'properties.bootstrap.servers' ='Service IP address of the Kafka Broker instance:Kafka port'
    ,'properties.group.id' = 'group-1'
    ,'value.format' = 'raw'
    ,'scan.startup.mode' = 'group-offsets'
    ,'properties.auto.offset.reset' = 'earliest'
    ,'value.fields-include' = 'EXCEPT_KEY'
);
create table print (
    id STRING,
    name STRING
) with (
      'connector' = 'print'
);
INSERT INTO print
SELECT
    JSON_VAL(msgValue, '$.id'),
    JSON_VAL(msgValue, '$.name')
from kafkaSource;
```

6.7.13 Reusing FlinkSQL Lookup Operator

This section applies to MRS 3.5.0 or later.

Scenarios

When the Lookup Join result is written to multiple sinks, you do not need to copy a Lookup Join operator for each sink. This improves job performance.

How to Use

When you configure a Flink job, set **table.optimizer.graph-merge-enabled** to **true** on the Flink job development page of the FlinkServer web UI to reuse the Lookup operator. For details, see **Creating a FlinkServer Job**.

The following is an example SQL statement:

```
create table hudimor (
  uuid varchar(20),
  name varchar(10),
  age int,
  ts timestamp
) with (
  'connector' = 'hudi',
  'table.type' = 'MERGE_ON_READ',
  'path' = 'hdfs:///tmp/hudimor',
  'lookup.cache' = 'ALL',
  'lookup.cache.ttl' = '60000',
  'lookup.cache.partitioned' = 'true',
  'lookup.parallelism' = '3'
);
CREATE TABLE datagen1 (
```



```
  uuid varchar(20),
  name varchar(10),
  age int,
  ts timestamp(6),
  proctime as PROCTIME()
) WITH (
  'connector' = 'datagen',
  'rows-per-second' = '5'
);
create view view1 as
select
  t1.uuid as uuid,
  t1.name as name,
  t1.age as age,
  t1.ts as ts
FROM
  datagen1 AS t1
  left JOIN hudimor FOR SYSTEM_TIME AS OF t1.proctime AS t2 ON t1.uuid = t2.uuid;
CREATE TABLE blackhole1 (uuid varchar(20), name varchar(10)) WITH ('connector' = 'blackhole');
CREATE TABLE blackhole2 (uuid varchar(20), age int) WITH ('connector' = 'blackhole');
insert into
  blackhole1
select
  uuid,
  name
from
  view1;
insert into
  blackhole2
select
  uuid,
  age
from
  view1;
```

6.7.14 FlinkSQL Function Enhancements

This section applies to MRS 3.5.0 or later.

DATE_ADD Function

The DATE_ADD function is used to return the date after a specified number of days are added to a date.

- Parameters
  - Date: The data type is TIMESTAMP or STRING (format: yyyy-MM-dd HH:mm:ss). The value can be NULL.
  - Number of days to add: The data type is INT.
  - Return date: The date obtained by adding the specified number of days to the date. The data type is STRING.
- Example SQL

```
CREATE TABLE source (
  time1 TIMESTAMP
) WITH (
  'connector' = 'datagen',
  'rows-per-second' = '1'
);
create table Sink (
  date1 string,
  date2 string,
  date3 string
) with ('connector' = 'print');
INSERT into
```



```
  Sink
select
  DATE_ADD(time1, 30) as date1,
  DATE_ADD('2017-09-15 00:00:00', 30) as date2,
  DATE_ADD(cast(null as timestamp),30) as date3
FROM source
```

Table 6-41 Return value

date1 (string)	date2 (string)	date3 (string)
2024-06-28	2017-10-15	null

DATE_SUB Function

The DATE_SUB function is used to return the date obtained by subtracting a specified number of days from a date.

- Parameter
  - Date: The data type is TIMESTAMP or STRING (format: yyyy-MM-dd HH:mm:ss). The value can be NULL.
  - Number of days to subtract: The data type is INT.
  - Return date: The date obtained by subtracting the specified number of days from the date. The data type is STRING.
- Example SQL

```
CREATE TABLE source (
  time1 TIMESTAMP
) WITH (
  'connector' = 'datagen',
  'rows-per-second' = '1'
);
create table Sink (
  date1 string,
  date2 string,
  date3 string
) with ('connector' = 'print');
INSERT into
  Sink
select
  DATE_SUB(time1,30) as date1,
  DATE_SUB('2017-09-15 00:00:00', 30) as date2,
  DATE_SUB(cast(null as timestamp),30) as date3
FROM source
```

Table 6-42 Return value

date1 (string)	date2 (string)	date3 (string)
2024-04-29	2017-08-16	null

6.7.15 Using the MultiJoin Operator in Flink SQL

This topic is available for MRS 3.5.0 or later only.

Joining wide tables with Flink's Full outer Join operator increases backend pressure and slows performance. You can use MultiJoin operator to combine wide tables, doubling the computing performance.

Restrictions on Using the MultiJoin Operator in Flink SQL

- The MultiJoin operator supports only FULL OUTER JOIN and INNER JOIN.
- The MultiJoin operator supports only equi-joins. For non-equi joins, you can use a view for filtering.
- When the MultiJoin operator is used to join multiple tables, the join keys of all tables must be the same.
- The MultiJoin operator supports only **table.exec.state.ttl** and does not support table-level TTL or JTL.
- States of Full outer join are incompatible with MultiJoin states. Jobs cannot be restored using snapshots if you switch between the two operators.

Using the MultiJoin Operator in Flink SQL

When you configure a Flink job, set **table.optimizer.multi-join-enabled** to **true** on the Flink job development page of the FlinkServer web UI to use the MultiJoin operator. For details, see **Creating a FlinkServer Job**.

The following is an example SQL statement:

```
CREATE TABLE datagen1 (
  id int,
  f1 int,
  PRIMARY KEY (id) NOT ENFORCED
) WITH (
  'connector' = 'datagen'
);

CREATE TABLE datagen2 (
  id int,
  f2 int,
  PRIMARY KEY (id) NOT ENFORCED
) WITH (
  'connector' = 'datagen'
);

CREATE TABLE datagen3 (
  id int,
  f3 int,
  PRIMARY KEY (id) NOT ENFORCED
) WITH (
  'connector' = 'datagen'
);

create view datagen3_view AS
select
  *
from
  datagen3
where
  id > 1;

CREATE TABLE print (
  id int,
  f1 int,
  f2 int,
  f3 int,
  PRIMARY KEY (id) NOT ENFORCED
) WITH (
```



```
  'connector' = 'print'
);

insert into print
select
  COALESCE(datagen1.id,datagen2.id,datagen3_view.id),
  datagen1.f1,
  datagen2.f2,
  datagen3_view.f3
from datagen1
Full outer join datagen2 on datagen1.id = datagen2.id
Full outer join datagen3_view on datagen1.id = datagen3_view.id;
```

6.8 Flink O&M Management

6.8.1 Typical Flink Configuration Parameters

Configuration Files

All Flink parameters are configurable on the client. You are advised to modify the **flink-conf.yaml** configuration file on the client. If Flink parameters are modified on FusionInsight Manager, you need to download and install the client again after the configuration is complete.

- Configuration file path: *Client installation path*/**Flink/flink/conf/flink-conf.yaml**
- File configuration format: *Key*:*Value*

  Example: **taskmanager.heap.size: 1024mb**

  A space is required between *Key*: and *Value*.

Configuration Parameter Types

- **JobManager & TaskManager**:

  JobManager and TaskManager are main components of Flink, which is used for different security and performance scenarios. The configuration items include communication ports, memory management, and connection retry.
- **Blob server**:

  The Blob server on the JobManager node is used to receive JAR files uploaded by users on the client, send JAR files to TaskManager, and transfer log files. The configuration items include ports, SSL, retries, and concurrency.
- **Distributed Coordination (via Akka)**:

  The Akka actor model is the basis of communications between a Flink client and JobManager, JobManager and TaskManager, as well as TaskManagers. Related parameters can be configured based on the network environment or optimization policy. The configuration items include the timeout settings for message sending and waiting and the Akka listening mechanism DeathWatch.
- **SSL**:

  To configure a secure Flink cluster, you need to configure SSL-related configuration items, including SSL, certificate, password, and encryption algorithm.

- **Network communication (via Netty)**:

  When Flink runs a job, data transmission and reverse pressure detection between tasks depend on Netty. In certain environments, **Netty** parameters should be configured. For advanced optimization, you can adjust some Netty configuration items. The default configuration can meet the requirements of concurrent and high-throughput tasks in a large-scale cluster.

- **JobManager Web Frontend**:

  When JobManager is started, the web server is started in the same process. You can access the web server to obtain information about the current Flink cluster, including JobManager, TaskManager, and jobs running in the cluster. Configuration items of the web server include the port, temporary directory, display items, error redirection, and security-related items.

- **File Systems**:

  When a task is running, a result file is created. You can configure the file creation behavior, including the file overwriting strategy and directory creation.

- **State Backend**:

  Flink enables HA and job exception, as well as job pause and recovery during version upgrade. Flink depends on state backend to store job states and on the restart strategy to restart a job. You can configure state backend and the restart strategy. Configuration items include the state backend type, storage path, and restart strategy.

- **Kerberos-based Security**:

  Kerberos-related configuration items must be configured in Flink's security mode. The configuration items include keytab and principal of Kerberos.

- **HA**:

  The HA mode of Flink depends on ZooKeeper. So, ZooKeeper configuration items must be configured, including the ZooKeeper address, path, and security authentication.

- **Environment**:

  In scenarios raising special requirements on JVM configuration, users can use configuration items to transfer JVM parameters to the client, JobManager, and TaskManager.

- **Yarn**:

  Flink runs on a Yarn cluster and JobManager runs on ApplicationMaster. Some configuration parameters of JobManager depend on YARN. You can configure YARN-related configurations to enable Flink to better run on YARN. The configuration items include the memory, virtual kernel, and port of YARN containers.

- **Pipeline**:

  The Netty connection is used among multiple jobs to reduce latency. In this case, NettySink is used on the server and NettySource is used on the client for data transmission. Configuration items include NettySink information storing path, range of NettySink listening port, whether to enable SSL encryption, domain of the network used for NettySink monitoring.

- **Enabling the Alarm Function for Job Submission on the Client**:

  By default, the alarm function is disabled for jobs submitted through the Flink client. To enable it, install two FlinkServer instances on the node where the

jobs are submitted and configure related parameters in the **flink-conf.yaml** file on the client.

JobManager & TaskManager

Table 6-43 JobManager & TaskManager parameters

Parameter	Description	Default Value	Mandatory
taskmanager.rpc.port	IPC port range of TaskManager	32326-32390	No
taskmanager.memory.segment-size	Size of the memory buffer used by the memory manager and network stack The unit is bytes.	32768	No
taskmanager.data.port	Data exchange port range of TaskManager	32391-32455	No
taskmanager.data.ssl.enabled	Whether to enable secure sockets layer (SSL) encryption for data transfer between TaskManagers. This parameter is valid only when the global switch **security.ssl** is enabled.	false	No
taskmanager.numberOfTaskSlots	Number of slots occupied by TaskManager. Generally, the value is configured as the number of cores of the physical machine. In **yarn-session** mode, the value can be transmitted by only the **-s** parameter. In **yarn-cluster** mode, the value can be transmitted by only the **-ys** parameter.	1	No
parallelism.default	Default degree of parallelism, which is used for jobs for which the degree of parallelism is not specified	1	No
task.cancellation.interval	Interval between two successive task cancellation attempts. The unit is millisecond.	30000	No
client.rpc.port	Akka system listening port on the Flink client.	32651-32720	No

Parameter	Description	Default Value	Mandatory
jobmanager.heap.size	Size of the heap memory of JobManager. In **yarn-session** mode, the value can be transmitted by only the **-jm** parameter. In **yarn-cluster** mode, the value can be transmitted by only the **-yjm** parameter. If the value is smaller than **yarn.scheduler.minimum-allocation-mb** in the Yarn configuration file, the Yarn configuration value is used. The unit is B/KB/MB/GB/TB.	1024mb	No
taskmanager.heap.size	Size of the heap memory of TaskManager. In **yarn-session** mode, the value can be transmitted by only the **-tm** parameter. In **yarn-cluster** mode, the value can be transmitted by only the **-ytm** parameter. If the value is smaller than **yarn.scheduler.minimum-allocation-mb** in the Yarn configuration file, the Yarn configuration value is used. The unit is B/KB/MB/GB/TB.	1024mb	No
taskmanager.network.numberOfBuffers	Number of TaskManager network transmission buffer stacks. If an error indicates insufficient system buffer, increase the parameter value.	2048	No
taskmanager.debug.memory.startLogThread	Enable this item for debugging Flink memory and garbage collection (GC)-related problems. TaskManager periodically collects memory and GC statistics, including the current utilization of heap and off-heap memory pools and GC time.	false	No
taskmanager.debug.memory.logIntervalMs	Interval at which TaskManager periodically collects memory and GC statistics.	0	No
taskmanager.maxRegistrationDuration	Maximum duration of TaskManager registration on JobManager. If the actual duration exceeds the value, TaskManager is disabled.	5 min	No

Parameter	Description	Default Value	Mandatory
taskmanager.initial-registration-pause	Initial interval between two consecutive registration attempts. The value must contain a time unit (ms/s/min/h/d), for example, 5 seconds. The time value and unit are separated by half-width spaces. ms/s/m/h/d indicates millisecond, second, minute, hour, and day, respectively.	500 ms	No
taskmanager.max-registration-pause	Maximum registration retry interval in case of TaskManager registration failures. The unit is ms/s/m/h/d.	30 s	No
taskmanager.refused-registration-pause	Retry interval when a TaskManager registration connection is rejected by JobManager. The unit is ms/s/m/h/d.	10 s	No
classloader.resolve-order	Class resolution policies defined when classes are loaded from user codes, which means whether to first check the user code JAR file (**child-first**) or the application class path (**parent-first**). The default setting indicates that the class is first loaded from the user code JAR file, which means that the user code JAR file can contain and load dependencies that are different from those used by Flink.	child-first	No
slot.idle.timeout	Timeout for a vacant slot in Slot Pool, in milliseconds.	50000	No
slot.request.timeout	Timeout for requesting a slot from Slot Pool, in milliseconds.	300000	No
task.cancellation.timeout	Timeout of task cancellation, in milliseconds. If a task cancellation times out, a fatal TaskManager error may occur. If this parameter is set to **0**, no error is reported when a task cancellation times out.	180000	No
taskmanager.network.detailed-metrics	Indicates whether to enable the detailed metrics monitoring of network queue lengths.	false	No

Parameter	Description	Default Value	Mandatory
taskmanager.network.memory.buffers-per-channel	Maximum number of network buffers used by each outgoing/incoming stream (sub-partition/input streams). In credit-based flow control, this indicates how many credits are in each input stream. It should be configured with at least 2 buffers to deliver good performance. One buffer is used to receive in-flight data in the sub-partition, and the other for parallel serialization.	2	No
taskmanager.network.memory.floating-buffers-per-gate	Number of extra network buffers used by each output gate (result partition) or input gate, indicating the amount of floating credit shared among all input channels in credit-based flow control mode. Floating buffers are distributed based on the backlog feedback (real-time output buffers in sub-partitions) and can help mitigate back pressure caused by unbalanced data distribution among sub-partitions. Increase this value if the round-trip time between nodes is long and/or the number of machines in the cluster is large.	8	No
taskmanager.network.memory.fraction	Ratio of JVM memory used for network buffers, which determines how many streaming data exchange channels a TaskManager can have at the same time and the extent of channel buffering. Increase this value or the values of **taskmanager.network.memory.min** and **taskmanager.network.memory.max** if the job is rejected or a warning indicating that the system does not have enough buffers is received. Note that the values of **taskmanager.network.memory.min** and **taskmanager.network.memory.max** may overwrite this value.	0.1	No
taskmanager.network.memory.max	Maximum memory size of the network buffer. The value must contain a unit (B/KB/MB/GB/TB).	1 GB	No

Parameter	Description	Default Value	Mandatory
taskmanager.network.memory.min	Minimum memory size of the network buffer. The value must contain a unit (B/KB/MB/GB/TB).	64 MB	No
taskmanager.network.request-backoff.initial	Minimum backoff for partition requests of input channels.	100	No
taskmanager.network.request-backoff.max	Maximum backoff for partition requests of input channels.	10000	No
taskmanager.registration.timeout	Timeout for TaskManager registration. TaskManager will be terminated if it is not successfully registered within the specified time. The value must contain a time unit (ms/s/min/h/d).	5 min	No
resourcemanager.taskmanager-timeout	Timeout interval for releasing an idle TaskManager, in milliseconds.	30000	No

Blob server

Table 6-44 Blog server parameters

Parameter	Description	Default Value	Mandatory
blob.server.port	Blob server port	32456-32520	No
blob.service.ssl.enabled	Indicates whether to enable the encryption for the blob transmission channel. This parameter is valid only when the global switch **security.ssl** is enabled.	true	Yes
blob.fetch.retries	Number of times that TaskManager tries to download blob files from JobManager.	50	No
blob.fetch.num-concurrent	Number of concurrent tasks for downloading blob files supported by JobManager.	50	No

Parameter	Description	Default Value	Mandatory
blob.fetch.backlog	Number of blob files, such as **.jar** files, to be downloaded in the queue supported by JobManager. The unit is count.	1000	No
library-cache-manager.cleanup.interval	Interval at which JobManager deletes the JAR files stored on the HDFS when the user cancels the Flink job. The unit is second.	3600	No

Distributed Coordination (via Akka)

Table 6-45 Distributed Coordination parameters

Parameter	Description	Default Value	Mandatory
akka.ask.timeout	Timeout duration of Akka asynchronous and block requests. If a Flink timeout failure occurs, this value can be increased. Timeout occurs when the machine processing speed is slow or the network is blocked. The unit is ms/s/m/h/d.	10s	No
akka.lookup.timeout	Timeout duration for JobManager actor object searching. The unit is ms/s/m/h/d.	10s	No
akka.framesize	Maximum size of the message transmitted between JobManager and TaskManager. If a Flink error occurs because the message exceeds this limit, the value can be increased. The unit is b/B/KB/MB.	10485760b	No
akka.watch.heartbeat.interval	Heartbeat interval at which the Akka DeathWatch mechanism detects disconnected TaskManager. If TaskManager is frequently and incorrectly marked as disconnected due to heartbeat loss or delay, the value can be increased. The unit is ms/s/m/h/d.	10s	No

Parameter	Description	Default Value	Mandatory
akka.watch.heartbeat.pause	Acceptable heartbeat pause for Akka DeathWatch mechanism. A small value indicates that irregular heartbeat is not accepted. The unit is ms/s/m/h/d.	60s	No
akka.watch.threshold	DeathWatch failure detection threshold. A small value may mark normal TaskManager as failed and a large value increases failure detection time.	12	No
akka.tcp.timeout	Timeout duration of Transmission Control Protocol (TCP) connection request. If TaskManager connection timeout occurs frequently due to the network congestion, the value can be increased. The unit is ms/s/m/h/d.	20s	No
akka.throughput	Number of messages processed by Akka in batches. After an operation, the processing thread is returned to the thread pool. A small value indicates the fair scheduling for actor message processing. A large value indicates improved overall performance but lowered scheduling fairness.	15	No
akka.log.lifecycle.events	Switch of Akka remote time logging, which can be enabled for debugging.	false	No
akka.startup-timeout	Timeout interval before a remote component fails to be started. The value must contain a time unit (ms/s/min/h/d).	The value is the same as the value of **akka.ask.timeout**.	No
akka.ssl.enabled	Switch of Akka communication SSL. This parameter is valid only when the global switch **security.ssl** is enabled.	true	Yes
akka.client-socket-worker-pool.pool-size-factor	Factor that is used to determine the thread pool size. The pool size is calculated based on the following formula: ceil (available processors * factor). The size is bounded by the **pool-size-min** and **pool-size-max** values.	1.0	No

Parameter	Description	Default Value	Mandatory
akka.client-socket-worker-pool.pool-size-max	Maximum number of threads calculated based on the factor.	2	No
akka.client-socket-worker-pool.pool-size-min	Minimum number of threads calculated based on the factor.	1	No
akka.client.timeout	Timeout duration of the client. The value must contain a time unit (ms/s/min/h/d).	60s	No
akka.server-socket-worker-pool.pool-size-factor	Factor that is used to determine the thread pool size. The pool size is calculated based on the following formula: ceil (available processors * factor). The size is bounded by the **pool-size-min** and **pool-size-max** values.	1.0	No
akka.server-socket-worker-pool.pool-size-max	Maximum number of threads calculated based on the factor.	2	No
akka.server-socket-worker-pool.pool-size-min	Minimum number of threads calculated based on the factor.	1	No

SSL

Table 6-46 SSL parameters

Parameter	Description	Default Value	Mandatory
security.ssl.protocol	SSL transmission protocol version.	TLSv1.2	Yes

Parameter	Description	Default Value	Mandatory
security.ssl.algorithms	Supported SSL standard algorithm. For details, visit the Java official website at: **http://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html#ciphersuites**.	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Yes
security.ssl.enabled	Whether to enable SSL for internal communication. This parameter is automatically configured based on the cluster installation mode.	• Security mode: **true** • Normal mode: **false**	Yes
security.ssl.keystore	Java keystore file.	-	Yes
security.ssl.keystore-password	Password used to decrypt the keystore file.	-	Yes
security.ssl.key-password	Password used to decrypt the server key in the keystore file.	-	Yes
security.ssl.truststore	**truststore** file containing the public CA certificates.	-	Yes
security.ssl.truststore-password	Password used to decrypt the truststore file.	-	Yes

Network communication (via Netty)

Table 6-47 Network communication parameters

Parameter	Description	Default Value	Mandatory
taskmanager.network.netty.num-arenas	Number of Netty memory blocks.	1	No

Parameter	Description	Default Value	Mandatory
taskmanager.network.netty.server.numThreads	Number of Netty server threads	1	No
taskmanager.network.netty.client.numThreads	Number of Netty client threads	1	No
taskmanager.network.netty.client.connectTimeoutSec	Netty client connection timeout duration. The unit is second.	120	No
taskmanager.network.netty.sendReceiveBufferSize	Size of Netty sending and receiving buffers. This defaults to the system buffer size (**cat /proc/sys/net/ipv4/tcp_[rw]mem**) and is 4 MB in modern Linux. The unit is bytes.	4096	No
taskmanager.network.netty.transport	Netty transport type, either **nio** or **epoll**	nio	No

JobManager Web Frontend

Table 6-48 JobManager Web Frontend parameters

Parameter	Description	Default Value	Mandatory
jobmanager.web.allow-access-address	Web access whitelist. IP addresses are separated by commas (,). Only whitelisted IP addresses can access the web.	*	Yes

Parameter	Description	Default Value	Mandatory
flink.security.enable	When installing a Flink cluster, you are required to select **security mode** or **normal mode**. • If **security mode** is selected, this parameter is automatically set to **true**. • If **normal mode** is selected, this parameter is automatically set to **false**. For an installed Flink cluster, you can view the configured value to determine whether the cluster is in security or normal mode.	Automatic configuration	No
rest.bind-port	The web port. Value range: 32261–32325.	32261-32325	No
jobmanager.web.history	Number of recent jobs to be displayed.	5	No
jobmanager.web.checkpoints.disable	Whether to disable checkpoint statistics.	false	No
jobmanager.web.checkpoints.history	Number of checkpoint statistical records.	10	No
jobmanager.web.backpressure.cleanup-interval	Interval for clearing unaccessed backpressure records. The unit is millisecond.	600000	No
jobmanager.web.backpressure.refresh-interval	Interval for updating backpressure records. The unit is millisecond.	60000	No
jobmanager.web.backpressure.num-samples	Number of stack tracing records for reverse pressure calculation.	100	No
jobmanager.web.backpressure.delay-between-samples	Sampling interval for reverse pressure calculation. The unit is millisecond.	50	No

Parameter	Description	Default Value	Mandatory
jobmanager.web.ssl.enabled	Whether SSL encryption is enabled for web transmission. This parameter is valid only when the global switch **security.ssl** is enabled.	false	Yes
jobmanager.web.accesslog.enable	Switch to enable or disable web operation logs. The log is stored in **webaccess.log**.	true	Yes
jobmanager.web.x-frame-options	Value of the HTTP security header **X-Frame-Options**. The value can be **SAMEORIGIN**, **DENY**, or **ALLOW-FROM uri**.	DENY	Yes
jobmanager.web.cache-directive	Whether the web page can be cached.	no-store	Yes
jobmanager.web.expires-time	Expiration duration of web page cache. The unit is millisecond.	0	Yes
jobmanager.web.access-control-allow-origin	Web page same-origin policy that prevents cross-domain attacks.	*	Yes
jobmanager.web.refresh-interval	Web page refresh interval. The unit is millisecond.	3000	Yes
jobmanager.web.logout-timer	Automatic logout interval when no operation is performed. The unit is millisecond.	600000	Yes
jobmanager.web.403-redirect-url	Web page access error 403. If 403 error occurs, the specified page will be redirected to.	Automatic configuration	Yes
jobmanager.web.404-redirect-url	Web page access error 404. If 404 error occurs, the specified page will be redirected to.	Automatic configuration	Yes
jobmanager.web.415-redirect-url	Web page access error 415. If 415 error occurs, the specified page will be redirected to.	Automatic configuration	Yes
jobmanager.web.500-redirect-url	Web page access error 500. If 500 error occurs, the specified page will be redirected to.	Automatic configuration	Yes

Parameter	Description	Default Value	Mandatory
rest.await-leader-timeout	Time of the client waiting for the leader address. The unit is millisecond.	30000	No
rest.client.max-content-length	Maximum content length that the client handles (unit: bytes).	104857600	No
rest.connection-timeout	Maximum time for the client to establish a TCP connection (unit: ms).	15000	No
rest.idleness-timeout	Maximum time for a connection to stay idle before failing (unit: ms).	300000	No
rest.retry.delay	The time that the client waits between retries (unit: ms).	3000	No
rest.retry.max-attempts	The number of retry times if a retrievable operator fails.	20	No
rest.server.max-content-length	Maximum content length that the server handles (unit: bytes).	104857600	No
rest.server.numThreads	Maximum number of threads for the asynchronous processing of requests.	4	No
web.timeout	Timeout for web monitor (unit: ms).	10000	No

File Systems

Table 6-49 File Systems parameters

Parameter	Description	Default Value	Mandatory
fs.overwrite-files	Whether to overwrite the existing file by default when the file is written.	false	No

Parameter	Description	Default Value	Mandatory
fs.output.always-create-directory	When the degree of parallelism (DOP) of file writing programs is greater than 1, a directory is created under the output file path and different result files (each parallel write program) are stored in the directory. ● If this parameter is set to **true**, a directory is created for the writing program whose DOP is 1 and a result file is stored in the directory. ● If this parameter is set to **false**, the file of the writing program whose DOP is 1 is created directly in the output path and no directory is created.	false	No

State Backend

Table 6-50 State Backend parameters

Parameter	Description	Default Value	Mandatory
state.backend.fs.checkpointdir	Path when the backend is set to **filesystem**. The path must be accessible by JobManager. Only the local mode is supported. In the cluster mode, use an HDFS path.	hdfs:///flink/checkpoints	No
state.savepoints.dir	Savepoint storage directory used by Flink to restore and update jobs. When a savepoint is triggered, the metadata of the savepoint is saved to this directory.	hdfs:///flink/savepoint	Mandatory in security mode
restart-strategy	Default restart policy, which is used for jobs for which no restart policy is specified. ● fixed-delay ● failure-rate ● none	none	No

Parameter	Description	Default Value	Mandatory
restart-strategy.fixed-delay.attempts	The retry times of the **fixed-delay** policy.	• If the checkpoint is enabled, the default value is the value of **Integer.MAX_VALUE**. • If the checkpoint is disabled, the default value is **3**.	No
restart-strategy.fixed-delay.delay	Retry interval when the fixed-delay strategy is used. The unit is ms/s/m/h/d.	• If the checkpoint is enabled, the default value is 10s. • If the checkpoint is disabled, the default value is the value of **akka.ask.timeout**.	No
restart-strategy.failure-rate.max-failures-per-interval	Maximum number of restart times in a specified period before a job fails when the fault rate policy is used.	1	No

Parameter	Description	Default Value	Mandatory
restart-strategy.failure-rate.failure-rate-interval	Retry interval when the failure-rate strategy is used. The unit is ms/s/m/h/d.	60 s	No
restart-strategy.failure-rate.delay	Retry interval when the failure-rate strategy is used. The unit is ms/s/m/h/d.	The default value is the same as the value of **akka.ask.timeout**. For details, see **Distributed Coordination (via Akka)**.	No

Kerberos-based Security

Table 6-51 Kerberos-based security parameters

Parameter	Description	Default Value	Mandatory
security.kerberos.login.keytab	Keytab file path. This parameter is a client parameter.	Configure the parameter based on actual service requirements.	Yes
security.kerberos.login.principal	A parameter on the client. If **security.kerberos.login.keytab** and **security.kerberos.login.principal** are both set, keytab certificate is used by default.	Configure the parameter based on actual service requirements.	No

Parameter	Description	Default Value	Mandatory
security.kerberos.login.contexts	Contexts of the jass file generated by Flink. This parameter is a server parameter.	Client, KafkaClient	Yes

HA

Table 6-52 HA parameters

Parameter	Description	Default Value	Mandatory
high-availability	Whether HA is enabled. Only the following two modes are supported currently: • none: Only a single JobManager is running. The checkpoint is disabled for JobManager. • ZooKeeper: – In non-Yarn mode, multiple JobManagers are supported and the leader JobManager is elected. – In Yarn mode, only one JobManager exists.	zookeeper	No
high-availability.zookeeper.quorum	ZooKeeper quorum address.	Automatic configuration	No
high-availability.zookeeper.path.root	Root directory that Flink creates on ZooKeeper, storing metadata required in HA mode.	/flink	No
high-availability.storageDir	Directory for storing JobManager metadata of state backend. ZooKeeper stores only pointers to actual data.	hdfs:///flink/recovery	No
high-availability.zookeeper.client.session-timeout	Session timeout duration on the ZooKeeper client. The unit is millisecond.	60000	No
high-availability.zookeeper.client.connection-timeout	Connection timeout duration on the ZooKeeper client. The unit is millisecond.	15000	No

Parameter	Description	Default Value	Mandatory
high-availability.zookeeper.client.retry-wait	Retry waiting time on the ZooKeeper client. The unit is millisecond.	5000	No
high-availability.zookeeper.client.max-retry-attempts	Maximum retry times on the ZooKeeper client.	3	No
high-availability.job.delay	Delay of job restart when JobManager recovers.	The default value is the same as the value of **akka.ask.timeout**.	No
high-availability.zookeeper.client.acl	Configure the ACL (open creator) of the ZooKeeper node. The ACL is automatically configured based on the security mode of the cluster.	• Security mode: The default value is **creator**. • Non-security mode: The default value is **open**.	Yes
zookeeper.sasl.disable	Whether to enable SASL authentication. This parameter is automatically configured based on the security mode of the cluster.	• Security mode: **false** • Non-security mode: **true**	Yes
zookeeper.sasl.service-name	• If the ZooKeeper server configures a service whose name is different from **ZooKeeper**, this configuration item can be set. • If service names on the client and server are inconsistent, authentication fails.	zookeeper	Yes

Environment

Table 6-53 Environment parameters

Parameter	Description	Default Value	Mandatory
env.java.opts	JVM parameter, which is transferred to the startup script, JobManager, TaskManager, and Yarn client. For example, transfer remote debugging parameters.	-Xloggc:<LOG_DIR>/gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=20M -Djdk.tls.ephemeralDHKeySize=2048 -Djava.library.path=${HADOOP_COMMON_HOME}/lib/native -Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false -Dbeetle.application.home.path=/opt/*xxx*/Bigdata/common/runtime/security/config	No

Yarn

Table 6-54 YARN parameters

Parameter	Description	Default Value	Mandatory
yarn.maximum-failed-containers	Maximum number of containers the system is going to reallocate in case of a container failure of TaskManager The default value is the number of TaskManagers when the Flink cluster is started.	5	No

Parameter	Description	Default Value	Mandatory
yarn.application-attempts	Number of ApplicationMaster restarts. The value is the maximum value in the validity interval that is set to Akka's timeout in Flink. After the restart, the IP address and port number of ApplicationMaster will change and you will need to connect to the client manually.	2	No
yarn.heartbeat-delay	Time between heartbeats with the ApplicationMaster and Yarn ResourceManager in seconds. The unit is second.	5	No
yarn.containers.vcores	Number of virtual cores of each Yarn container	Number of TaskManager slots	No
yarn.application-master.port	ApplicationMaster port number setting. A port number range is supported.	32586-32650	No

Pipeline

Table 6-55 Pipeline parameters

Parameter	Description	Default Value	Mandatory
nettyconnector.registerserver.topic.storage	Path (on a third-party server) to information about IP address, port numbers, and concurrency of NettySink. ZooKeeper is recommended for storage.	/flink/nettyconnector	No. However, if pipeline is enabled, the feature is mandatory.
nettyconnector.sinkserver.port.range	Port range of NettySink.	28444-28843	No. However, if pipeline is enabled, the feature is mandatory.

Parameter	Description	Default Value	Mandatory
nettyconnector.ssl.enabled	Whether SSL encryption for the communication between NettySink and NettySource is enabled. For details about the encryption key and protocol, see **SSL**.	false	No. However, if pipeline is enabled, the feature is mandatory.
nettyconnector.message.delimiter	Delimiter used to configure the message sent by NettySink to the NettySource, which is 2-4 bytes long, and cannot contain **\n**, **#**, or space.	The default value is **$_**.	No. However, if pipeline is enabled, the feature is mandatory.

Enabling the Alarm Function for Job Submission on the Client

This function applies to MRS 3.2.0 or later.

Table 6-56 Parameters for enabling the alarm function for job submission on the client

Parameter	Description	Value	Mandatory
job.alarm.enable	Whether to enable the alarm function.	true	Yes
flinkserver.host.ip	Service IP addresses of the two FlinkServer instances	*x.x.x.x,x.x.x.x*	Yes

6.8.2 Interconnecting Flink with AOM

This topic is available for MRS 3.5.0 and later versions only.

Scenarios

Application Operations Management (AOM) is an an observable platform and provides integrated monitoring capabilities based on metrics, links, logs, and events. Flink can send monitoring indicators to AOM via Prometheus instances, making it easy to view monitored data.

This example describes how to interconnect Flink with AOM using FlinkResource, FlinkServer, or a client and upload Flink monitoring metrics to AOM.

Prerequisites

- AOM has been enabled.
- The HDFS, YARN, Kafka, and Flink components have been installed in the MRS cluster.
- The client that contains the Flink service has been installed, for example, in the /opt/client directory.

Procedure

Step 1 Create a Prometheus instance in AOM.

1. Log in to the AOM console.
2. In the navigation pane on the left, choose **Prometheus Monitoring** > **Instances**. On the displayed page, click **Add Prometheus Instance**.
3. Enter an instance name, specify the enterprise project and instance type, and click **OK**.

Table 6-57 Parameters for creating a Prometheus instance

Parameter	Description
Instance Name	Prometheus instance name. Enter a maximum of 100 characters and do not start or end with an underscore (_) or hyphen (-). Only letters, digits, underscores, and hyphens are allowed.
Enterprise Project	Enterprise project name. – If you have selected **ALL** for **Enterprise Project** on the global settings page, select an enterprise project from the drop-down list. – If you have already selected an enterprise project on the global settings page, this option will be grayed and cannot be changed.
Instance Type	Type of the Prometheus instance. Select **Common Prometheus Instance**.

4. In the instance list, click the name of the created Prometheus instance you just created. Click **Settings** in the navigation pane, and check and save the following information.

Table 6-58 Prometheus instance information

Type	Parameter	Example Data
Credential	AppSecret	00000001000000001070C6A*xx*0E2EF73

Type	Parameter	Example Data
Service Address	URL in the Configuration Code for Prometheus Remote Write pane in the **Intranet** tab	https://aom-internal-access.*xxx*/push

Step 2 Configure parameters for interconnecting with AOM on MRS.

1. Log in to FusionInsight Manager.
2. Choose **Cluster** > **Services** > **Flink** > **Configurations** > **All Configurations**.
3. Configure parameters for Flink to interconnect with AOM.
   - Interconnect with AOM through FlinkServer.

     Choose **FlinkServer(Role)** > **Customization**, add parameters in **Table 6-59** to **flink.customized.configs**, save the configuration, and restart involved FlinkServer instances.

     Table 6-59 Parameters for interconnecting Flink with AOM

Parameter Name	Value	Mandatory	Description
metrics.reporter	alarm,aom	Yes	**alarm** means FlinkServer alarms for customized parameters overwriting original Flink parameters. Set both **alarm** and **aom**.
metrics.reporter.aom.url	URL obtained from **Step 1.4**	Yes	URL for connecting to the AOM Prometheus instance.
metrics.reporter.aom.access.code	Value of AppSecret obtained from **Step 1.4**	Yes	Credential for calling the AOM Prometheus instance.
metrics.reporter.aom.factory.class	com.huawei.mrs.flink.AomMetricReporterFactroy	Yes	Implementation class for reporting Flink metrics. The fixed value is **com.huawei.mrs.flink.AomMetricReporterFactroy**
metrics.reporter.aom.interval	30s	Yes	Interval for reporting Flink metrics, in seconds.

Parameter Name	Value	Mandatory	Description
metrics.reporter.aom.version	0.1.0	No	AOM version information.
metrics.reporter.aom.filterLabelValue-Characters	true	No	Whether to filter the label value. The content that cannot be matched by [a-zA-Z0-9:_] will be filtered. ■ **true** (default value): Enable the filtering function. ■ **false**: Disable the filtering function.
metrics.reporter.aom.jobName	-	No	Name of the job. The name will be sent to the Prometheus instance as a label value.
metrics.reporter.aom.randomJobNameSuffix	true	No	Whether to add a random suffix to the job name. ■ **true** (default value): Add a random suffix. ■ **false**: Do not add a random suffix.
metrics.reporter.aom.groupingKey	-	No	Grouping key, which is the group and global label of all metrics. The value is in the format of k1=v1,k2=v2.

- Interconnect with AOM through FlinkResource.
  i. Choose **FlinkResource(Role)** > **Customization**, add parameters in **Table 6-59** to **flink.customized.configs**, save the configuration, and restart involved FlinkResource instances.
  ii. Reinstall the client or update the existing client configuration.
- Interconnect with AOM through the Flink client.
  i. Log in to the node where the client is installed as the client installation user.
  ii. Add the parameters in **Table 6-59** to the **/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml** file and save the file.

Step 3 View monitoring metrics.

After the Flink job is successfully executed, log in to the AOM 2.0 console. In the navigation pane on the left, choose **Prometheus Monitoring** > **Instances**. Click

the name of the Prometheus instance created in **Step 1**. View the monitoring information on the **Metric Management** page.

----End

6.8.3 Flink Log Overview

Log Description

Log path:

- Run logs of a Flink job: **${BIGDATA_DATA_HOME}/hadoop/data${i}/nm/containerlogs/application_${appid}/container_{$contid}**

  NOTE

  The logs of executing tasks are stored in the preceding path. After the execution is complete, the Yarn configuration determines whether these logs are gathered to the HDFS directory.

- FlinkResource run logs: **/var/log/Bigdata/flink/flinkResource**
- Run logs related to FlinkServer HA scripts (MRS 3.2.0 or later): **/var/log/Bigdata/audit/flink/flinkserver/ha**

Log archive rules:

1. FlinkResource run logs:
   - By default, service logs are backed up each time when the log size reaches 20 MB. A maximum of 20 logs can be reserved without being compressed.
   - You can set the log size and number of compressed logs on the Manager page or modify the corresponding configuration items in **log4j-cli.properties**, **log4j.properties**, and **log4j-session.properties** in *Client installation directory*/**Flink/flink/conf/** on the client.

   Table 6-60 FlinkResource log list

Type	Name	Description
FlinkResource run logs	checkService.log	Health check log
	kinit.log	Initialization log
	postinstall.log	Service installation log
	prestart.log	Prestart script log
	start.log	Startup log

2. FlinkServer service logs, HA-related logs, and audit logs.
   - By default, FlinkServer service logs, HA-related logs, and audit logs are backed up each time when the log size reaches 100 MB. The service logs are stored for a maximum of 30 days, and audit logs are stored for a maximum of 90 days.
   - You can set the log size and number of compressed logs on the Manager page or modify the corresponding configuration items in **log4j-**

cli.properties, **log4j.properties**, and **log4j-session.properties** in *Client installation directory*/**Flink/flink/conf/** on the client.

Table 6-61 FlinkServer log list

Type	Name	Description
FlinkServer run logs	checkService.log	Health check log
	checkFlinkServer.log	Health check log of FlinkServer
	localhost_access_log..*yyyy-mm-dd*.txt	URL log of FlinkServer
	start_thrift_server.out	Thrift server startup log
	thrift_server_thriftServer_*xxx*.log.last	
	cleanup.log	Cleanup log file for instance installation and uninstallation
	flink-omm-client-*IP*.log	Job startup log
	flinkserver_*yyyymmdd-x*.log.gz	Service archive log
	flinkserver.log	Service log
	flinkserver---*pidxxxx*-gc.log.*x*.current	GC log
	kinit.log	Initialization log
	postinstall.log	Service installation log
	prestart.log	Prestart script log
	start.log	Startup log
	stop.log	Stop log
	catalina.*yyyy-mm-dd*.log	Tomcat run log
	catalina.out	
	host-manager.*yyyy-mm-dd*.log	
	localhost.*yyyy-mm-dd*.log	
	manager.*yyyy-mm-dd*.log	

Type	Name	Description
Run log file related to FlinkServer HA scripts (MRS 3.2.0 or later)	ha.log	HA run log
	ha_monitor.log	HA process monitoring log
	floatip_ha.log	Floating IP address resource script log
	rcommflinkserver.log	FlinkServer resource script log
	checkHaStatus.log	HA process log
	checknode.log	HA health status log
	rs-sendAlarm.log	HA alarm sending log
	flink_roll.log	FlinkServer active/standby switchover log (active/standby switchover required)
FlinkServer audit logs	flinkserver_audit_*yyyymmdd-x*.log.gz	Audit archive log
	flinkserver_audit.log	Audit log
Stack information log (MRS 3.2.0 or later)	threadDump-<*DATE*>.log	Log printed when instances are restarted or stopped

Log Level

Table 6-62 describes the log levels supported by Flink. The priorities of log levels are ERROR, WARN, INFO, and DEBUG in descending order. Logs whose levels are higher than or equal to the specified level are printed. The number of printed logs decreases as the specified log level increases.

Table 6-62 Log levels

Level	Description
ERROR	Error information about the current event processing
WARN	Exception information about the current event processing
INFO	Normal running status information about the system and events
DEBUG	System information and system debugging information

To modify log levels, perform the following steps:

Step 1 Go to the **All Configurations** page of Flink by referring to **Modifying Cluster Service Configuration Parameters**.

Step 2 On the menu bar on the left, select the log menu of the target role.

Step 3 Select a desired log level.

Step 4 Save the configuration. In the displayed dialog box, click **OK** to make the configurations take effect.

----End

NOTE

- After the configuration is complete, you do not need to restart the service. Download the client again for the configuration to take effect.
- You can also change the configuration items corresponding to the log level in **log4j-cli.properties**, **log4j.properties**, and **log4j-session.properties** in *Client installation directory*/**Flink/flink/conf/** on the client.
- When a job is submitted using a client, a log file is generated in the **log** folder on the client. The default umask value is **0022**. Therefore, the default log permission is **644**. To change the file permission, you need to change the umask value. For example, to change the umask value of user **omm**:
  - Add **umask 0026** to the end of the **/home/omm/.baskrc** file.
  - Run the **source /home/omm/.baskrc** command to make the file permission take effect.

Log Format

Table 6-63 Log formats

Type	Format	Example
Run log	*<yyyy-MM-dd HH:mm:ss,SSS>*|*<Log level>*|*<Name of the thread that generates the log>*|*<Message in the log>*|*<Location where the log event occurs>*	2019-06-27 21:30:31,778 | INFO | [flink-akka.actor.default-dispatcher-3] | TaskManager container_e10_1498290698388_0004_02_000007 has started. | org.apache.flink.yarn.YarnFlinkResourceManager (FlinkResourceManager.java:368)

6.9 Flink Performance Tuning

6.9.1 Flink Memory GC Optimization Parameters

Scenarios

The computing of Flink depends on memory. If the memory is insufficient, the performance of Flink will be greatly deteriorated. One solution is to monitor

garbage collection (GC) to evaluate the memory usage. If the memory becomes the performance bottleneck, optimize the memory usage according to the actual situation.

If **Full GC** is frequently reported in the Container GC on the Yarn that monitors the node processes, the GC needs to be optimized.

NOTE

In the **env.java.opts** configuration item of the **conf/flink-conf.yaml** file on the client, add the **-Xloggc:<LOG_DIR>/gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=20M** parameter. The GC log is configured by default.

Procedure

- Optimize GC.

  Adjust the ratio of tenured generation memory to young generation memory. In the **conf/flink-conf.yaml** configuration file on the client, add the **-XX:NewRatio** parameter to the **env.java.opts** configuration item. For example, **-XX:NewRatio=2** indicates that ratio of tenured generation memory to young generation memory is 2:1, that is, the young generation memory occupies one third and tenured generation memory occupies two thirds.
- When developing Flink applications, optimize the partitioning or grouping operation of DataStream.
  - If partitioning causes data skew, partitions need to be optimized.
  - Do not perform concurrent operations, because some operations, WindowAll for example, to DataStream do not support parallelism.
  - Do not use set keyBy to string type.

6.9.2 Flink Job Concurrency

Scenario

The degree of parallelism (DOP) indicates the number of tasks to be executed concurrently. It determines the number of data blocks after the operation. Configuring the DOP will optimize the number of tasks, data volume of each task, and the host processing capability.

Query the CPU and memory usage. If data and tasks are not evenly distributed among nodes, increase the DOP for even distribution.

Procedure

Configure the DOP at one of the following layers (the priorities of which are in the descending order) based on the actual memory, CPU, data, and application logic conditions:

- Operator

  Call the **setParallelism()** method to specify the DOP of an operator, data source, and sink. For example:

```
final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
```



```
DataStream<String> text = [...]
DataStream<Tuple2<String, Integer>> wordCounts = text
    .flatMap(new LineSplitter())
    .keyBy(0)
    .timeWindow(Time.seconds(5))
    .sum(1).setParallelism(5);

wordCounts.print();

env.execute("Word Count Example");
```

- Execution environment

  Flink runs in the execution environment which defines a default DOP for operators, data source and data sink.

  Call the **setParallelism()** method to specify the default DOP of the execution environment. Example:

  ```
  final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
  env.setParallelism(3);
  DataStream<String> text = [...]
  DataStream<Tuple2<String, Integer>> wordCounts = [...]
  wordCounts.print();
  env.execute("Word Count Example");
  ```

- Client

  Specify the DOP when submitting jobs to Flink on the client. If you use the CLI client, specify the DOP using the **-p** parameter. Example:

  ```
  ./bin/flink run -p 10 ../examples/*WordCount-java*.jar
  ```

- System

  On the Flink client, modify the **parallelism.default** parameter in the **flink-conf.yaml** file under the conf to specify the DOP for all execution environments.

6.9.3 Flink Job Process Parameters

Scenario

In Flink on Yarn mode, there are JobManagers and TaskManagers. JobManagers and TaskManagers schedule and run tasks.

Therefore, configuring parameters of JobManagers and TaskManagers can optimize the execution performance of a Flink application. Perform the following steps to optimize the Flink cluster performance.

Procedure

Step 1 Configure JobManager memory.

JobManagers are responsible for task scheduling and message communications between TaskManagers and ResourceManagers. JobManager memory needs to be increased as the number of tasks and the DOP increases.

JobManager memory needs to be configured based on the number of tasks.

- When running the **yarn-session** command, add the **-jm MEM** parameter to configure the memory.
- When running the **yarn-cluster** command, add the **-yjm MEM** parameter to configure the memory.

Step 2 Configure the number of TaskManagers.

Each core of a TaskManager can run a task at the same time. Increasing the number of TaskManagers has the same effect as increasing the DOP. Therefore, you can increase the number of TaskManagers to improve efficiency when there are sufficient resources.

Step 3 Configure the number of TaskManager slots.

Multiple cores of a TaskManager can process multiple tasks at the same time. This has the same effect as increasing the DOP. However, the balance between the number of cores and the memory must be maintained, because all cores of a TaskManager share the memory.

- When running the **yarn-session** command, add the **-s NUM** parameter to configure the number of slots.
- When running the **yarn-cluster** command, add the **-ys NUM** parameter to configure the number of slots.

Step 4 Configure TaskManager memory.

TaskManager memory is used for task execution and communication. A large-size task requires more resources. In this case, you can increase the memory.

- When running the **yarn-session** command, add the **-tm MEM** parameter to configure the memory.
- When running the **yarn-cluster** command, add the **-ytm MEM** parameter to configure the memory.

----End

6.9.4 Flink Netty Network Communication Parameters

Scenarios

The communication of Flink is based on Netty network. The network performance determines the data switching speed and task execution efficiency. Therefore, the performance of Flink can be optimized by optimizing the Netty network.

Procedure

In the **conf/flink-conf.yaml** file on the client, change configurations as required. Exercise caution when changing default values, because default values are optimal.

- **taskmanager.network.netty.num-arenas**: Specifies the number of arenas of Netty. The default value is**taskmanager.numberOfTaskSlots**.
- **taskmanager.network.netty.server.numThreads**and**taskmanager.network.netty.client.numThreads**: specify the number of threads on the client and server. The default value is**taskmanager.numberOfTaskSlots**.
- **taskmanager.network.netty.client.connectTimeoutSec**: specifies the timeout interval for connection of TaskManager client. The default value is**120s**.

- **taskmanager.network.netty.sendReceiveBufferSize**: specifies the buffer size of the Netty network. The default value is the buffer size (cat /proc/sys/net/ipv4/tcp_[rw]mem) of the system and the value is usually 4 MB.
- **taskmanager.network.netty.transport**: specifies the transmission method of the Netty network. The default value is**nio**. The value can only be**nio**and**epoll**.

6.9.5 RocksDB State Backend of Flink Jobs

This topic is available for MRS 3.3.0 and later versions only.

Flink Job RocksDB

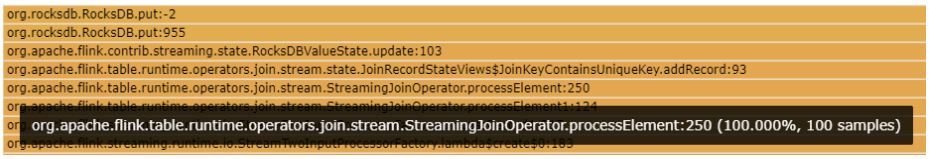
When RocksDB is enabled as the state backend for jobs, a large amount of state data causes poor read and write performance of RocksDB. You can perform the following operations to check whether the operator performance is affected by RocksDB:

- On the ThreadDump of TaskManager, check whether the operator has been executed on the RocksDB operation interface for a long time. If the following information is displayed after multiple refreshes, the operator is executed on the RocksDB operation interface for a long time.

```
Join[5] -> Calc[6] -> Sink: print[7] (1/1)#0" Id=113 RUNNABLE (in native)
    at org.rocksdb.RocksDB.put(Native Method)
    at org.rocksdb.RocksDB.put(RocksDB.java:955)
    at org.apache.flink.contrib.streaming.state.RocksDBValueState.update(RocksDBValueState.java:103)
```

- Enable the flame graph (set **rest.flamegraph.enabled** to **true**) and submit the job again to view operator hotspots. Operator hotspots reach 100% in the figure below.

  Figure 6-17 Viewing operator hotspots in a flame graph



When the RockDB read/write latency is long, you can enable RocksDB monitoring and alarm reporting to optimize RockDB parameters based on the monitoring and alarm items. After job optimization, you are advised to disable RockDB monitoring and alarm reporting because they will deteriorate RocksDB performance by 5% to 10%.

To avoid impact on other jobs, RocksDB monitoring is configured by setting user-defined parameters. This section describes how to enable RocksDB monitoring, alarm reporting, and optimization parameters.

Enabling RocksDB Monitoring for Flink Jobs

Step 1 Log in to FusionInsight Manager as a user with the FlinkServer administrator rights.

Step 2 Choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link next to **Flink WebUI** to access the Flink web UI.

Step 3 Click **Job Management**. The job management page is displayed.

Step 4 Locate the job that is to be optimized and is not in the **Running** state, and click **Develop** in the **Operation** column to go to the job development page.

Step 5 In the **Custom Configuration** area on the job development page, add the following parameters and save the settings:

- Enabling RocksDB monitoring

Table 6-64 RocksDB monitoring configuration

Parameter	Value	Description
state.backend.rocksdb.metrics.hot.enabled	true	Non-statistical monitoring of RocksDB includes the monitoring items contained in RocksDB Property.
state.backend.rocksdb.metrics.statistics.enabled	true	RocksDB monitoring statistics
state.backend.rocksdb.metrics.num-immutable-mem-table	true	Monitors the number of immutable memtables in RocksDB. If the value keeps increasing or exceeds the threshold, the write performance will be affected.
state.backend.rocksdb.metrics.mem-table-flush-pending	true	Monitors the number of pending memtable flushes in RocksDB.
state.backend.rocksdb.metrics.compaction-pending	true	Monitors the number of pending compactions in RocksDB. If there are any pending compactions, **1** is returned. Otherwise, **0** is returned.
state.backend.rocksdb.metrics.background-errors	true	Monitors the number of metrics.background-errors in RocksDB.
state.backend.rocksdb.metrics.cur-size-active-mem-table	true	Monitors the approximate size of the active memtable, in bytes.
state.backend.rocksdb.metrics.cur-size-all-mem-tables	true	Monitors the approximate size of active and unflushed immutable memtables, in bytes.
state.backend.rocksdb.metrics.size-all-mem-tables	true	Monitors the approximate size of active, unflushed, and pinned memtables, in bytes.
state.backend.rocksdb.metrics.num-entries-active-mem-table	true	Monitors the total number of entries in active memtables.

Parameter	Value	Description
state.backend.rocksdb.metrics.num-entries-imm-mem-tables	true	Monitors the total number of entries in immutable memtables.
state.backend.rocksdb.metrics.num-deletes-active-mem-table	true	Monitors the total number of deleted entries in active memtables.
state.backend.rocksdb.metrics.num-deletes-imm-mem-tables	true	Monitors the total number of deleted entries in unflushed immutable memtables.
state.backend.rocksdb.metrics.estimate-num-keys	true	Monitors the number of keys in RocksDB.
state.backend.rocksdb.metrics.estimate-table-readers-mem	true	Monitors the memory used to read SST tables, excluding the memory used in the block cache (such as filters and index blocks), in bytes.
state.backend.rocksdb.metrics.num-snapshots	true	Monitors the number of unpublished snapshots in the database.
state.backend.rocksdb.metrics.num-live-versions	true	Monitors the number of real-time versions. A version is an internal data schema. If there are too many versions, RocksDB may fail to delete old versions due to query or compaction operations.
state.backend.rocksdb.metrics.estimate-live-data-size	true	Monitors the real-time data volume, in bytes (usually smaller than the size of an SST file due to space amplification).
state.backend.rocksdb.metrics.total-sst-files-size	true	Monitors the total size of SST files of all versions, in bytes. Too many files may slow down query.
state.backend.rocksdb.metrics.live-sst-files-size	true	Monitors the total size of all SST files of the latest version, in bytes. Too many files may slow down query.
state.backend.rocksdb.metrics.estimate-pending-compaction-bytes	true	Monitors the total size of compaction data, in bytes. This ensures that the size of compaction data at all levels is smaller than the target size, and other compactions beyond the levels are invalid.
state.backend.rocksdb.metrics.num-running-compactions	true	Monitors the number of running compactions. If all threads are in the **Running** state, the write performance may be affected.

Parameter	Value	Description
state.backend.rocksdb.metrics.num-running-flushes	true	Monitors the number of running flush tasks. If all threads are in the **Running** state, the write performance may be affected.
state.backend.rocksdb.metrics.actual-delayed-write-rate	true	Monitors the actual delayed write rate. If **0** is returned, there is no delay.
state.backend.rocksdb.metrics.is-write-stopped	true	Monitors whether write to RocksDB is stopped. If the write is stopped, **1** is returned. Otherwise, **0** is returned.
state.backend.rocksdb.metrics.block-cache-capacity	true	Monitors the block cache capacity.
state.backend.rocksdb.metrics.block-cache-usage	true	Monitors the memory occupied by data in the block cache.
state.backend.rocksdb.metrics.block-cache-pinned-usage	true	Monitors the memory occupied by pinned data in the block cache.
state.backend.rocksdb.metrics.compression-ratio	true	Monitors the compression ratio of each layer.
state.backend.rocksdb.metrics.compression-ratio-levelN	7	Number of layers whose compression ratio is to be monitored. The value must be at least 0 and not greater than the configured number of layers.
state.backend.rocksdb.metrics.num-files	true	Monitors the number of files at each layer.
state.backend.rocksdb.metrics.num-files-levelN	7	Number of layers whose file quantity is to be monitored. The value must be at least 0 and not greater than the configured number of layers.

Parameter	Value	Description
state.backend.rocksdb.metrics.statistics.ticker	block.cache.miss,block.cache.hit,block.cache.index.miss,block.cache.index.hit,block.cache.filter.miss,block.cache.filter.hit,block.cache.data.miss,block.cache.data.hit,bloom.filter.useful,memtable.hit,memtable.miss,l0.hit,l1.hit,l2andup.hit,stall.micros	Statistics ticker monitoring item To add a monitoring item, add it to the end of the value and separate monitoring items with commas (,).
state.backend.rocksdb.metrics.statistics.histogram	db.get.micros,db.write.micros,db.flush.micros,compaction.times.micros	Statistics straight square monitoring item To add a monitoring item, add it to the end of the value and separate monitoring items with commas (,).

- Enabling the RocksDB alarm function

Table 6-65 RocksDB alarm configuration

Configuration Item	Default Value	Description
metrics.reporter.alarm.job.alarm.rocksdb.metrics.enable	true	Whether to enable RocksDB monitoring. This function is disabled by default. This configuration is valid only when the state backend is RocksDB.
metrics.reporter.alarm.job.alarm.rocksdb.metrics.duration	180s	Interval for RocksDB to monitor alarms
metrics.reporter.alarm.job.alarm.rocksdb.metrics.print.enabled	true	Whether to print RocksDB monitoring information to TaskManager If **metrics.reporter.alarm.job.alarm.rocksdb.metrics.enable** is set to **true**, this parameter is automatically set to **true** by default.
metrics.reporter.alarm.job.alarm.rocksdb.metrics.print.interval	5min	Interval for printing RocksDB monitoring information to TaskManager
metrics.reporter.alarm.job.alarm.rocksdb.get.micros.threshold	1000	Time threshold of the Get operation, in µs. If the time consumed by a job exceeds the threshold consecutively within the period specified by **metrics.reporter.alarm.job.alarm.rocksdb.metrics.duration**, an alarm is reported.
metrics.reporter.alarm.job.alarm.rocksdb.write.micros.threshold	3000	Time threshold of the Write operation, in µs. If the time consumed by a job exceeds the threshold consecutively within the period specified by **metrics.reporter.alarm.job.alarm.rocksdb.metrics.duration**, an alarm is reported.
metrics.reporter.alarm.job.alarm.actual-delayed-write-rate.threshold	0	If the write rate of a job is limited consecutively within the period specified by **metrics.reporter.alarm.job.alarm.rocksdb.metrics.duration**, an alarm is reported. The value **0** indicates that the write rate is not limited.
metrics.reporter.alarm.job.alarm.rocksdb.background.jobs.multiplier	2	An alarm is reported when the number of flush or compaction requests exceeds the multiplier of **state.backend.rocksdb.thread.num**.

Step 6 On the **Job Management** page, click **Start** to run the job. Based on the RocksDB monitoring and alarm information, add the following parameters in the **Custom Parameters** area on the job development page to optimize the job. After job optimization is complete, you are advised to disable RocksDB monitoring and alarm reporting.

Table 6-66 RocksDB optimization parameters

Parameter	Value	Description
state.backend.rocksdb.writebuffer.count	2	Number of active and immutable memtables. If the write speed is too fast or the number of Flink threads is too small, the write is blocked. When **SPINNING_DISK_OPTIMIZED_HIGH_MEM** is enabled, the default value is **4**. It is recommended that the value be greater than or equal to the value of **state.backend.rocksdb.writebuffer.number-to-merge** plus 2.
state.backend.rocksdb.writebuffer.size	64MB	Memtable size
state.backend.rocksdb.thread.num	2	Number of RocksDB flush and compaction threads. When **SPINNING_DISK_OPTIMIZED_HIGH_MEM** is enabled, the default value is **4**.
state.backend.rocksdb.writebuffer.number-to-merge	1	Number of immutable flushes. Deduplication is performed when *n* immutable flushes are performed. When **SPINNING_DISK_OPTIMIZED_HIGH_MEM** is enabled, the default value is **3**.
state.backend.rocksdb.compaction.level.max-size-level-base	256MB	Total size of SSL files at level 1. When **SPINNING_DISK_OPTIMIZED_HIGH_MEM** is enabled, the default value is **1 GB**.
state.backend.rocksdb.compaction.level.target-file-size-base	64MB	Size of SSL files at level 1+. When **SPINNING_DISK_OPTIMIZED_HIGH_MEM** is enabled, the default value is **128 MB**.
state.backend.rocksdb.num_levels	7	Number of RocksDB levels
state.backend.rocksdb.level0_slowdown_writes_trigger	20	Number of files that trigger slowdown at level 0. If the value is smaller than 0, slowdown will never be triggered.
state.backend.rocksdb.level0_stop_writes_trigger	36	Maximum number of files that trigger stop at level 0

Parameter	Value	Description
state.backend.rocksdb.max_compaction_bytes	-	Maximum number of bytes in a compaction. The default value is the **state.backend.rocksdb.compaction.level.target-file-size-base** value x 25.
state.backend.rocksdb.level0_file_num_compaction_trigger	4	Compaction from level 0 to level 1 is triggered when the number of Level 0 SSTs reaches the threshold.
state.backend.rocksdb.compression	snappy	SST file compression algorithm The value can be **null**, **snapp**, **zlib**, **bzip2**, **lz4**, **lz4hc**, **xpress**, or **zstd**.
state.backend.rocksdb.bottommost_compression	snappy	The bottom layer uses heavyweight compression types to reduce space. The underlying data may be cold data. To enable this function, you are advised to use **zstd** or **zlib**. The value can be **null**, **snapp**, **zlib**, **bzip2**, **lz4**, **lz4hc**, **xpress**, or **zstd**.
state.backend.rocksdb.max_bytes_for_level_multiplier	10	Data volume multiplier factor of level 1 plus two adjacent layers
state.backend.rocksdb.hard-pending-compaction-bytes-limit	256GB	When the pending compaction size exceeds the threshold, write operations are stopped.
state.backend.rocksdb.soft-pending-compaction-bytes-limit	64GB	When the pending compaction size exceeds the threshold, the write traffic is limited.
state.backend.rocksdb.use-bloom-filter	true	Bloom filter. After this function is enabled, each newly created SST file contains a Bloom filter.
state.backend.rocksdb.block.cache-size	8MB	Cache size. When **SPINNING_DISK_OPTIMIZED_HIGH_MEM** is enabled, the default value is **256MB**.
state.backend.rocksdb.block.blocksize	4KB	Block size. When **SPINNING_DISK_OPTIMIZED_HIGH_MEM** is enabled, the default value is **128KB**.
state.backend.rocksdb.files.open	-1	Maximum number of opened handles, which is mainly used for SST file handles. The value **-1** indicates that the number is not limited.

----End

6.9.6 Separate Storage of Cold and Hot Data for Flink Job State Backend

This topic is available for MRS 3.3.0 or later only.

In wide table joins, each table contains a large number of fields. State backends hold a large volume of data and the processing speed is severely lowered. To solve this problem, you can enable tiered storage for hot-cold separation.

Procedure

Step 1 Install the client that contains services such as Flink and HBase. The installation path is for example, **/opt/hadoopclient**.

Step 2 Log in to the node where the client is installed as the client installation user and copy all configuration files in the **/opt/client/HBase/hbase/conf/** directory of HBase to an empty directory of all nodes where FlinkServer is deployed, for example, **/tmp/client/HBase/hbase/conf/**.

Change the owner of the configuration file directory and its upper-layer directory on the FlinkServer node to **omm**.

chown omm: /tmp/client/HBase/ -R

NOTE

- FlinkServer nodes:

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Flink** > **Instances**, and check the **Service IP Address** of FlinkServer.
- If the node where a FlinkServer instance is deployed is the node where the HBase client is installed, skip this step on this node.

Step 3 Log in to Manager and choose **Cluster** > **Services** > **Flink**. Click **Configurations** then **All Configurations**, search for the **HBASE_CONF_DIR** parameter, and enter the FlinkServer directory (for example, **/tmp/client/HBase/hbase/conf/**) to which the HBase configuration files are copied in **Step 2** in **Value**.

Step 4 After the parameters are configured, click **Save**. After confirming the modification, click **OK**.

Step 5 Click **Instances**, select all FlinkServer instances, choose **More** > **Restart Instance**, enter the password, and click **OK** to restart the instances.

Step 6 Log in to FusionInsight Manager as a user with the FlinkServer administrator rights.

Step 7 Choose **Cluster** > **Services** > **Flink**. In the **Basic Information** area, click the link next to **Flink WebUI** to access the Flink web UI.

Step 8 Click **Job Management**. The job management page is displayed.

Step 9 Locate the job that is to be optimized and is not in the **Running** state, and click **Develop** in the **Operation** column to go to the job development page.

Step 10 In the **Custom Parameters** area on the **Job Development** page, add the following parameters as required and save the settings. For details about hot data (regularly used data), see **Table 6-67**. For details about cold data (data that is not required often), see **Table 6-68**.

Table 6-67 RocksDB state backend storage

Parameter	Description	Example Value
table.exec.state.cold.enabled	Whether to enable RocksDB that stores hot and cold data separately • **false** (default value): Periodic dynamic scaling is disabled. • **true**: Periodic dynamic scaling is enabled.	false
state.backend.rocksdb.cold.localdir	Directory for storing cold data	-
state.backend.rocksdb.cold.predefined-options	Predefined configuration of cold data RocksDB: • **DEFAULT** (default value): RocksDB disk is not written forcibly. You are advised to use this value. • **SPINNING_DISK_OPTIMIZED_HIGH_MEM**: Parameters for optimizing RocksDB disk write. Flink job recovery does not depend on RocksDB, so you are not advised to use the current configuration.	DEFAULT
state.backend	State backend storage medium. Set this parameter to **rocksdb**.	rocksdb

Table 6-68 HBase serves as the state backend storage for level-2 cold data

Parameter	Description	Example Value
table.exec.state.cold.enabled	Whether to enable tiered storage for hot and cold data • **false** (default value): Periodic dynamic scaling is disabled. • **true**: Periodic dynamic scaling is enabled.	false
state.backend.cold	State backend storage for cold data. Currently, only **hbase** is supported.	hbase

Parameter	Description	Example Value
table.exec.state.ttl	Timeout interval for data status changes • If **table.exec.state.cold.enabled** is **true**, this parameter indicates when hot data expires. When hot data is stored longer than the value, it becomes cold data. • If **table.exec.state.cold.enabled** is **false**, all expired data will be deleted. • Default value: 0, indicating that the data never expires.	0
state.backend.hbase.zookeeper.quorum	ZooKeeper connection address used to access HBase. Format: *Service IP address of the ZooKeeper quorumpeer instance*:*ZooKeeper client port* ,*Service IP address of the ZooKeeper quorumpeer instance*:*ZooKeeper client port* ,*Service IP address of the ZooKeeper quorumpeer instance*:*ZooKeeper client port*	192.168.10.10:24002,192.168.10.11:24002,192.168.10.12:24002
state.backend	State backend storage medium. Set this parameter to **rocksdb**.	rocksdb

NOTE

- IP address of the ZooKeeper quorumpeer instance

  To obtain IP addresses of all ZooKeeper quorumpeer instances, log in to FusionInsight Manager and choose **Cluster** > **Services** > **ZooKeeper**. On the displayed page, click **Instance** and view the IP addresses of all the hosts where the quorumpeer instances locate.
- Port number of the ZooKeeper client

  Log in to FusionInsight Manager and choose **Cluster** > **Service** > **ZooKeeper**. On the displayed page, click **Configurations** and check the value of **clientPort**.

----End

6.10 Typical Commands of the Flink Client

Before you use the Flink shell script, perform the following operations. For details, see **Using the Flink Client** to run a wordcount job.

Step 1 The Flink client has been installed, for example, in the **/opt/client** directory.

Step 2 Run the following command to initialize environment variables:

source /opt/client/bigdata_env

Step 3 If Kerberos authentication has been enabled for the cluster, configure client authentication by referring to **Step 5**. If Kerberos authentication is disabled, skip this step.

Step 4 Run the related commands according to **Table 6-69**.

Table 6-69 Flink Shell commands

Command	Description	Description
yarn-session.sh	**-at,--applicationType <arg>**: Defines the Yarn application type. **-D <property=value>**: Configures dynamic parameter. **-d,--detached**: Disables the interactive mode and starts a separate Flink Yarn session. **-h,--help**: Displays the help information about the Yarn session CLI. **-id,--applicationId <arg>**: Binds to a running Yarn session. **-j,--jar <arg>**: Sets the path of the user's JAR file. **-jm,--jobManagerMemory <arg>**: Sets the JobManager memory. **-m,--jobmanager <arg>**: Address of the JobManager (master) to which to connect. Use this parameter to connect to a specified JobManager. **-nl,--nodeLabel <arg>**: Specifies the nodeLabel of the Yarn application. **-nm,--name <arg>**: Customizes a name for the application on Yarn. **-q,--query**: Queries available Yarn resources. **-qu,--queue <arg>**: Specifies a Yarn queue. **-s,--slots <arg>**: Sets the number of slots for each TaskManager. **-t,--ship <arg>**: specifies the directory of the file to be sent. **-tm,--taskManagerMemory <arg>**: sets the TaskManager memory. **-yd,--yarndetached**: starts Yarn in the detached mode. **-z,--zookeeperNamespace <args>**: specifies the namespace of ZooKeeper. **-h**: Gets help information.	Start a resident Flink cluster to receive tasks from the Flink client.

Command	Description	Description
flink run	**-c,--class <classname>**: Specifies a class as the entry for running programs. **-C,--classpath <url>**: Specifies **classpath**. **-d,--detached**: Runs a job in the detached mode. **-files,--dependencyFiles <arg>**: File on which the Flink program depends. This parameter is available only in MRS 3.2.0 and later versions. **-n,--allowNonRestoredState**: A state that cannot be restored can be skipped during restoration from a snapshot point in time. For example, if an operator in the program is deleted, you need to add this parameter when restoring the snapshot point. **-m,--jobmanager <host:port>**: Specifies the JobManager. **-p,--parallelism <parallelism>**: Specifies the job DOP, which will overwrite the DOP parameter in the configuration file. **-q,--sysoutLogging**: Disables the function of outputting Flink logs to the console. **-s,--fromSavepoint <savepointPath>**: Specifies a savepoint path for recovering jobs. **-z,--zookeeperNamespace <zookeeperNamespace>**: specifies the namespace of ZooKeeper. **-yat,--yarnapplicationType <arg>**: Defines the Yarn application type. **-yD <arg>**: Dynamic parameter configuration. **-yd,--yarndetached**: Starts Yarn in the detached mode. **-yh,--yarnhelp**: Obtains the Yarn help. **-yid,--yarnapplicationId <arg>**: Binds a job to a Yarn session. **-yj,--yarnjar <arg>**: Sets the path to Flink jar file. **-yjm,--yarnjobManagerMemory <arg>**: Sets the JobManager memory (MB). **-ynm,--yarnname <arg>**: Customizes a name for the application on Yarn. **-yq,--yarnquery**: Queries available Yarn resources (memory and CPUs).	Submit a Flink job. 1. The **-y*** parameter is used in the **yarn-cluster** mode. 2. If the parameter is not **-y***, you need to run the **yarn-session** command to start the Flink cluster before running this command to submit a task.

Command	Description	Description
	-yqu,--yarnqueue <arg>: Specifies a Yarn queue. **-ys,--yarnslots**: Sets the number of slots for each TaskManager. **-yt,--yarnship <arg>**: Specifies the path of the file to be sent. **-ytm,--yarntaskManagerMemory <arg>**: Sets the TaskManager memory (MB). **-yz,--yarnzookeeperNamespace <arg>**: Specifies the namespace of ZooKeeper. The value must be the same as the value of **yarn-session.sh -z**. **-h**: Gets help information.	
flink info	**-c,--class <classname>**: Specifies a class as the entry for running programs. **-p,--parallelism <parallelism>**: Specifies the DOP for running programs. **-h**: Gets help information.	Display the execution plan (JSON) of the running program.
flink list	**-a,--all**: displays all jobs. **-m,--jobmanager <host:port>**: specifies the JobManager. **-r,--running:** displays only jobs in the running state. **-s,--scheduled**: displays only jobs in the scheduled state. **-z,--zookeeperNamespace <zookeeperNamespace>**: specifies the namespace of ZooKeeper. **-yid,--yarnapplicationId <arg>**: binds a job to a Yarn session. **-h**: gets help information.	Query running programs in the cluster.

Command	Description	Description
flink stop	**-d,--drain**: sends MAX_WATERMARK before the savepoint is triggered and the job is stopped. **-p,--savepointPath <savepointPath>**: path for storing savepoints. The default value is **state.savepoints.dir**. **-m,--jobmanager <host:port>**: specifies the JobManager. **-z,--zookeeperNamespace <zookeeperNamespace>**: specifies the namespace of ZooKeeper. **-yid,--yarnapplicationId <arg>**: binds a job to a Yarn session. **-h**: gets help information.	Forcibly stop a running job (only streaming jobs are supported. **StoppableFunction** needs to be implemented on the source side in service code).
flink cancel	**-m,--jobmanager <host:port>**: specifies the JobManager. **-s,--withSavepoint <targetDirectory>**: triggers a savepoint when a job is canceled. The default directory is **state.savepoints.dir**. **-z,--zookeeperNamespace <zookeeperNamespace>**: specifies the namespace of ZooKeeper. **-yid,--yarnapplicationId <arg>**: binds a job to a Yarn session. **-h**: gets help information.	Cancel a running job.
flink savepoint	**-d,--dispose <arg>**: specifies a directory for storing the savepoint. **-m,--jobmanager <host:port>**: specifies the JobManager. **-z,--zookeeperNamespace <zookeeperNamespace>**: specifies the namespace of ZooKeeper. **-yid,--yarnapplicationId <arg>**: binds a job to a Yarn session. **-h**: gets help information.	Trigger a savepoint.

Command	Description	Description
source *Client installation directory*/**bigdata_env**	None	Import client environment variables. Restriction: If the user uses a custom script (for example, **A.sh**) and runs this command in the script, variables cannot be imported to the **A.sh** script. If variables need to be imported to the custom script **A.sh**, the user needs to use the secondary calling method. For example, first call the **B.sh** script in the **A.sh** script, and then run this command in the **B.sh** script. Parameters can be imported to the **A.sh** script but cannot be imported to the **B.sh** script.
start-scala-shell.sh	local | remote <host> <port> | yarn: running mode	Start the scala shell.
sh generate_keystore.sh	-	Run the **generate_keystore.sh** script to generate security cookie, **flink.keystore**, and **flink.truststore**. You need to enter a user-defined password that does not contain number signs (#).

----End

6.11 Common Flink SQL Syntax

SELECT and WHERE

Filter using the WHERE clause.

- Syntax:

  SELECT *select_list* **FROM** *table_expression* **[WHERE** *boolean_expression* **]**

- Example:

```
SELECT price + tax FROM Orders WHERE id = 10
```

WITH

WITH provides a way to write auxiliary statements for use in a larger query. These statements, which are often referred to as Common Table Expression (CTE), can be thought of as defining temporary views that exist just for one query.

- Syntax:

  WITH <*with_item_definition***> [, ...]**

  SELECT ... FROM ...;

  <with_item_defintion>:

  with_item_name (*column_name***[, ...n]) AS (** *<select_query>* **)**

- Example:

  Define a common table expression **orders_with_total** and use it in a GROUP BY query.

```
WITH orders_with_total AS (
    SELECT order_id, price + tax AS total
    FROM Orders
)
SELECT order_id, SUM(total)
FROM orders_with_total
GROUP BY order_id;
```

Window Aggregation

Queries with a group by window aggregation will compute a single result row per group.

- Syntax:

  SELECT ...

  FROM *<windowed_table>* **-- relation applied windowing TVF**

  GROUP BY *window_start, window_end, …*

- Example:

```
SELECT window_start, window_end, SUM(price)
FROM TABLE(
TUMBLE(TABLE Bid, DESCRIPTOR(bidtime), INTERVAL '10' MINUTES))
GROUP BY window_start, window_end;
```


Window Deduplication

Window Deduplication is a special deduplication which removes rows that duplicate over a set of columns, keeping the first one or the last one for each window and partitioned keys.

- Syntax:

  SELECT *[column_list]*

  FROM (

  SELECT *[column_list]*,

  ROW_NUMBER() OVER (PARTITION BY *window_start, window_end [, col_key1...]*

  ORDER BY *time_attr [asc|desc])* **AS** *rownum*

  FROM *table_name)* **-- relation applied windowing TVF**

  WHERE *(rownum = 1 | rownum <=1 | rownum < 2)* **[AND** *conditions***]**

- Example:

```
SELECT *
FROM (
SELECT bidtime, price, item, supplier_id, window_start, window_end,
ROW_NUMBER() OVER (PARTITION BY window_start, window_end ORDER BY bidtime DESC) AS
rownum
FROM TABLE(
TUMBLE(TABLE Bid, DESCRIPTOR(bidtime), INTERVAL '10' MINUTES))
) WHERE rownum <= 1;
```

Top-N

Top-N queries ask for the N smallest or largest values ordered by columns.

- Syntax:

  SELECT *[column_list]*

  FROM (

  SELECT *[column_list],*

  ROW_NUMBER() OVER *([PARTITION BY col1[, col2...]]*

  ORDER BY *col1 [asc|desc][, col2 [asc|desc]...])* **AS** *rownum*

  FROM *table_name)*

  WHERE *rownum* **<=** *N* **[AND** *conditions***]**

- Example:

```
CREATE TABLE ShopSales (
product_id   STRING,
category     STRING,
product_name STRING,
sales        BIGINT
) WITH (...);

SELECT *
FROM (
SELECT *,
ROW_NUMBER() OVER (PARTITION BY category ORDER BY sales DESC) AS row_num
FROM ShopSales)
WHERE row_num <= 5
```


6.12 Common Issues About Flink

Avoiding Data Skew

If data skew occurs (certain data volume is extremely large), the execution time of tasks is inconsistent even though no GC is performed.

- Redefine keys. Use keys of smaller granularity to optimize the task size.
- Modify the DOP.
- Call the rebalance operation to balance data partitions.

Setting Timeout Interval for the Buffer

- During the execution of tasks, data is exchanged through network. You can set the **setBufferTimeout** parameter to specify a buffer timeout interval for data exchanging among different servers.
- If **setBufferTimeout** is set to **-1**, the refreshing operation is performed when the buffer is full to maximize the throughput. If **setBufferTimeout** is set to **0**, the refreshing operation is performed each time data is received to minimize the delay. If **setBufferTimeout** is set to a value greater than **0**, the refreshing operation is performed after the buffer times out.

  The following is an example:

```
env.setBufferTimeout(timeoutMillis);

env.generateSequence(1,10).map(new MyMapper()).setBufferTimeout(timeoutMillis);
```

Reserving Resources

Reserve certain Yarn resources in the cluster for other tasks. For example, assume that there are 100 vCPU cores and 200 GB memory. Take 90 vCPU cores and 180 GB, and reserve about 10% of the total resources for automatic task retry and recovery in case of node faults.

6.13 Flink Troubleshooting

Running the yarn-session Command Fails to Create a Flink Cluster When a Different User Is Used

There are two users with the same permissions: **testuser** and **bdpuser**. When user **testuser** is used to create a Flink cluster, no error message is displayed. While user **bdpuser** is used to create a Flink cluster, an error message is displayed during the **yarn-session.sh** command execution.

```
2019-01-02 14:28:09,098 | ERROR | [main] | Ensure path threw exception |
org.apache.flink.shaded.curator.org.apache.curator.framework.imps.CuratorFrameworkImpl
(CuratorFrameworkImpl.java:566)
org.apache.flink.shaded.zookeeper.org.apache.zookeeper.KeeperException$NoAuthException:
KeeperErrorCode = NoAuth for /flink/application_1545397824912_0022
```

This is because that the HA configuration item is not modified. In the Flink configuration file, the default value of **high-availability.zookeeper.client.acl** is

creator, indicating that only the creator has the access permission. A new user cannot access the directory on ZooKeeper. As a result, the **yarn-session.sh** command execution fails.

Perform the following steps to solve the problem:

1. Modify the value of **high-availability.zookeeper.path.root** in the **conf/flink-conf.yaml** file. For example, run the following command:
   ```
   high-availability.zookeeper.path.root: flink2
   ```
2. Submit the Flink job again.

Error Message "security.kerberos.login.keytab" Is Displayed When a Command Is Executed on the Flink Client

The client was successfully installed. When you run a client command, for example, **yarn-session.sh**, an error message is displayed, as shown in the following figure.

```
[root@host01 bin]# yarn-session.sh
2018-10-25 01:22:06,454 | ERROR | [main] | Error while trying to split key and value in configuration file /opt/flinkclient/Flink/flink/conf/flink-conf.yaml:80: "security.kerberos.login.keytab: " | org.apache.flink.configuration.GlobalConfiguration (GlobalConfiguration.java:160)
Exception in thread "main" org.apache.flink.configuration.IllegalConfigurationException: Error while parsing YAML configuration file :80: "security.kerberos.login.keytab: "
```

In a secure cluster environment, Flink requires security authentication. The security authentication is not configured on the current client.

1. Flink uses the following two authentication modes:
   - Kerberos authentication: used by Flink yarn client, Yarn Resource Manager, JobManager, HDFS, TaskManager, Kafka, and ZooKeeper
   - Internal authentication mechanism of Yarn: used between Yarn ResourceManager and ApplicationMaster
2. If security mode is enabled for a cluster, you must use the Kerberos authentication and security cookie authentication. As shown in the logs, it is found that the **security.kerberos.login.keytab** setting in the configuration file is incorrect and the security configuration is not performed.

Perform the following steps to solve the problem:

1. Download the user **keytab** file from MRS and save it to a directory on the node where the Flink client is deployed.
2. Configure the following parameters in the **flink-conf.yaml** file:
   a. Keytab path
      ```
      security.kerberos.login.keytab: /home/flinkuser/keytab/abc222.keytab
      ```
      NOTE
      - **/home/flinkuser/keytab/abc222.keytab** indicates the user directory storing the keytab file in **1**.
      - You need to ensure that the client user has the permission on the corresponding directory.

   b. Principal name
      ```
      security.kerberos.login.principal: abc222
      ```
   c. In HA mode, if ZooKeeper is configured, the Kerberos authentication configuration items must be configured.


```
zookeeper.sasl.disable: false
security.kerberos.login.contexts: Client
```

d. If you want to perform Kerberos authentication between Kafka client and Kafka broker, set the value as follows:

```
security.kerberos.login.contexts: Client,KafkaClient
```


7 Using Flume

7.1 Flume Log Collection Overview

Flume is a distributed, reliable, and highly available system for aggregating massive logs, which can efficiently collect, aggregate, and move massive log data from different data sources and store the data in a centralized data storage system. Various data senders can be customized in the system to collect data. Additionally, Flume provides simple data processes capabilities and writes data to data receivers (which is customizable).

Flume consists of the client and server, both of which are FlumeAgents. The server corresponds to the FlumeServer instance and is directly deployed in a cluster. The client can be deployed inside or outside the cluster. he client-side and service-side FlumeAgents work independently and provide the same functions.

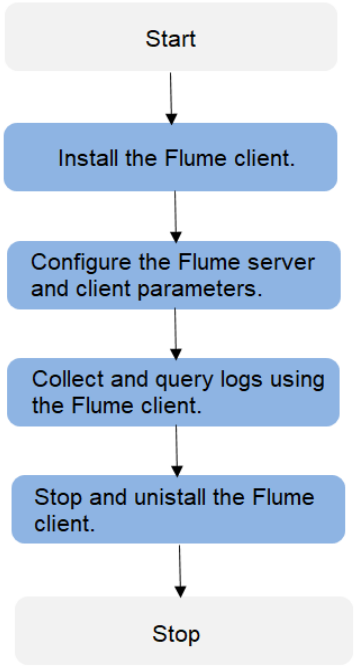
The client-side FlumeAgent needs to be independently installed. Data can be directly imported to components such as HDFS and Kafka. Additionally, the client-side and service-side FlumeAgents can also work together to provide services.

Process

Create a Flume cascading task using both Flume server and client. Perform the following operations to collect logs:

1. Installing the flume client
2. Configuring the Flume server and client parameters
3. Collecting and querying logs using the Flume client
4. Stopping and uninstalling the Flume client

Figure 7-1 Log collection process



Flume Modules

The Flume client or server consists of one or multiple agents. Each agent consists of three modules: source, channel, and sink. Data enters the source module, is transmitted to the channel, and then is sent to the sink for the next agent or a destination outside the client. **Table 7-1** describes Flume modules.

Table 7-1 Module description

Name	Description
Source	A source receives or generates data and sends the data to one or multiple channels. The source can work in either data-driven or polling mode. Typical sources include: • Sources that are integrated with the system and receives data, such as Syslog and Netcat • Sources that automatically generate event data, such as Exec and SEQ • IPC sources that are used for communication between agents, such as Avro A Source must associate with at least one channel.

Name	Description
Channel	A channel is used to buffer data between a source and a sink. After the sink transmits the data to the next channel or the destination, the cache is deleted automatically. The persistency of the channels varies with the channel types: ● Memory channel: non-persistency ● File channel: persistency implemented based on write-ahead logging (WAL) ● JDBC channel: persistency implemented based on the embedded database Channels support the transaction feature to ensure simple sequential operations. A channel can work with sources and sinks of any quantity.
Sink	Sink is responsible for sending data to the next hop or final destination and removing the data from the channel after successfully sending the data. Typical sinks include: ● Sinks that send storage data to the final destination, such as HDFS and Kafka ● Sinks that are consumed automatically, such as Null Sink ● IPC sinks that are used for communication between agents, such as Avro A sink must associate with at least one channel.

Each Flume agent can be configured with multiple source, channel, and sink modules. That is, one source sends data to multiple channels, and multiple sinks send the data to the next agent or destinations.

Multiple Flumes can be cascaded, meaning that the sink of an agent sends data to the source of another agent.

Supplementary Information

1. Flume provides the following reliability measures:
   - The transaction mechanism is implemented between sources and channels, and between channels and sinks.
   - The sink processor supports the failover and load balancing (load_balance) mechanisms.

     The following is an example of the load balancing (load_balance) configuration:

     ```
     server.sinkgroups=g1
     server.sinkgroups.g1.sinks=k1 k2
     server.sinkgroups.g1.processor.type=load_balance
     server.sinkgroups.g1.processor.backoff=true
     server.sinkgroups.g1.processor.selector=random
     ```
2. The following are precautions for the aggregation and cascading of multiple Flume clients:

  - Avro or Thrift protocol can be used for cascading.
  - When the aggregation end contains multiple nodes, evenly distribute the clients to these nodes. Do not connect all the clients to a single node.

3. The Flume client can contain multiple independent data flows. That is, multiple sources, channels, and sinks can be configured in the **properties.properties** configuration file. These components can be linked to form multiple flows.

   For example, to configure two data flows in a configuration, run the following commands:

```
server.sources = source1 source2
server.sinks = sink1 sink2
server.channels = channel1 channel2

#dataflow1
server.sources.source1.channels = channel1
server.sinks.sink1.channel = channel1

#dataflow2
server.sources.source2.channels = channel2
server.sinks.sink2.channel = channel2
```

7.2 Flume Service Model Configuration

Service Model Configuration Guide

Guide a reasonable Flume service configuration by providing performance differences between Flume common modules, to avoid a nonstandard overall service performance caused when a frontend Source and a backend Sink do not match in performance.

Only single channels are compared for description.

During Flume service configuration and module selection, the ultimate throughput of a sink must be greater than the maximum throughput of a source. Otherwise, in extreme load scenarios, the write speed of the source to a channel is greater than the read speed of sink from channel. Therefore, the channel is fully occupied due to frequent usage, and the performance is affected.

Avro Source and Avro Sink are usually used in pairs to transfer data between multiple Flume Agents. Therefore, Avro Source and Avro Sink do not become a performance bottleneck in general scenarios.

Inter-Module Performance

Based on comparison between the limit performances of modules, Kafka Sink and HDFS Sink can meet the throughput requirements when the front-end is SpoolDir Source. However, HBase Sink could become performance bottlenecks due to the low write performances thereof. As a result, data is stacked in Channel. If you have to use HBase Sink or other sinks that are prone to become performance bottlenecks, you can use **Channel Selector** or **Sink Group** to meet performance requirements.

Channel Selector

A channel selector allows a source to connect to multiple channels. Data of the source can be distributed or copied by selecting different types of selectors.

Currently, a channel selector provided by Flume can be a replicating channel selector or a multiplexing channel selector.

Replicating: indicates that the data of the source is synchronized to all channels.

Multiplexing: indicates that based on the value of a specific field of the header of an event, a channel is selected to send the data. In this way, the data is distributed based on a service type.

- Replicating configuration example:

```
client.sources = kafkasource
client.channels = channel1 channel2
client.sources.kafkasource.type = org.apache.flume.source.kafka.KafkaSource
client.sources.kafkasource.kafka.topics = topic1,topic2
client.sources.kafkasource.kafka.consumer.group.id = flume
client.sources.kafkasource.kafka.bootstrap.servers = 10.69.112.108:21007
client.sources.kafkasource.kafka.security.protocol = SASL_PLAINTEXT
client.sources.kafkasource.batchDurationMillis = 1000
client.sources.kafkasource.batchSize = 800
client.sources.kafkasource.channels = channel1 channel2

client.sources.kafkasource.selector.type = replicating
client.sources.kafkasource.selector.optional = channel2
```

Table 7-2 Parameters in the Replicating configuration example

Parameter	Default Value	Description
Selector.type	replicating	Selector type. Set this parameter to **replicating**.
Selector.optional	-	Optional channel. Configure this parameter as a list.

- Multiplexing configuration example:

```
client.sources = kafkasource
client.channels = channel1 channel2
client.sources.kafkasource.type = org.apache.flume.source.kafka.KafkaSource
client.sources.kafkasource.kafka.topics = topic1,topic2
client.sources.kafkasource.kafka.consumer.group.id = flume
client.sources.kafkasource.kafka.bootstrap.servers = 10.69.112.108:21007
client.sources.kafkasource.kafka.security.protocol = SASL_PLAINTEXT
client.sources.kafkasource.batchDurationMillis = 1000
client.sources.kafkasource.batchSize = 800
client.sources.kafkasource.channels = channel1 channel2

client.sources.kafkasource.selector.type = multiplexing
client.sources.kafkasource.selector.header = myheader
client.sources.kafkasource.selector.mapping.topic1 = channel1
client.sources.kafkasource.selector.mapping.topic2 = channel2
client.sources.kafkasource.selector.default = channel1
```

Table 7-3 Parameters in the Multiplexing configuration example

Parameter	Default Value	Description
Selector.type	replicating	Selector type. Set this parameter to **multiplexing**.

Parameter	Default Value	Description
Selector.header	Flume.selector.header	-

In a multiplexing selector example, select a field whose name is topic from the header of the event. When the value of the topic field in the header is topic1, send the event to a channel 1; or when the value of the topic field in the header is topic2, send the event to a channel 2.

Selectors need to use a specific header of an event in a source to select a channel, and need to select a proper header based on a service scenario to distribute data.

Sink Group

When the performance of a backend single sink is insufficient, and high reliability or heterogeneous output is required, you can use a sink group to connect a specified channel to multiple sinks, thereby meeting use requirements. Currently, Flume provides two types of sink processors to manage sinks in a sink group. The types are load balancing and failover.

Failover: Indicates that there is only one active sink in the sink group each time, and the other sinks are on standby and inactive. When the active sink becomes faulty, one of the inactive sinks is selected based on priorities to take over services, so as to ensure that data is not lost. This is used in high-reliability scenarios.

Load balancing: Indicates that all sinks in the sink group are active. Each sink obtains data from the channel and processes the data. In addition, during running, loads of all sinks in the sink group are balanced. This is used in performance improvement scenarios.

- Load balancing configuration examples:

```
client.sources = source1
client.sinks = sink1 sink2
client.channels = channel1

client.sinkgroups = g1
client.sinkgroups.g1.sinks = sink1 sink2
client.sinkgroups.g1.processor.type = load_balance
client.sinkgroups.g1.processor.backoff = true
client.sinkgroups.g1.processor.selector = random

client.sinks.sink1.type = logger
client.sinks.sink1.channel = channel1

client.sinks.sink2.type = logger
client.sinks.sink2.channel = channel1
```

Table 7-4 Parameters of Load Balancing configuration examples

Parameter	Default Value	Description
sinks	-	Specifies the sink list of the sink group. Multiple sinks are separated by spaces.

Parameter	Default Value	Description
processor.type	default	Specifies the type of a processor. Set this parameter to **load_balance**.
processor.backoff	false	Indicates whether to back off failed sinks exponentially.
processor.selector	round_robin	Specifies the selection mechanism. It must be round_robin, random, or a customized class that inherits AbstractSinkSelector.
processor.selector.maxTimeOut	30000	Specifies the time for masking a faulty sink. The default value is 30,000 ms.

- Failover configuration examples:

```
client.sources = source1
client.sinks = sink1 sink2
client.channels = channel1

client.sinkgroups = g1
client.sinkgroups.g1.sinks = sink1 sink2
client.sinkgroups.g1.processor.type = failover
client.sinkgroups.g1.processor.priority.sink1 = 10
client.sinkgroups.g1.processor.priority.sink2 = 5
client.sinkgroups.g1.processor.maxpenalty = 10000

client.sinks.sink1.type = logger
client.sinks.sink1.channel = channel1

client.sinks.sink2.type = logger
client.sinks.sink2.channel = channel1
```

Table 7-5 Parameters in the **failover** configuration example

Parameter	Default Value	Description
sinks	-	Specifies the sink list of the sink group. Multiple sinks are separated by spaces.
processor.type	default	Specifies the type of a processor. Set this parameter to **failover**.

Parameter	Default Value	Description
processor.priority.<sinkName>	-	Priority. **<sinkName>** must be defined in description of sinks. A sink having a higher priority is activated earlier. A larger value indicates a higher priority. **Note**: If there are multiple sinks, their priorities must be different. Otherwise, only one of them takes effect.
processor.maxpenalty	30000	Specifies the maximum backoff time of failed sinks (unit: ms).

Interceptors

The Flume interceptor supports modification or discarding of basic unit events during data transmission. You can specify the class name list of built-in interceptors in Flume or develop customized interceptors to modify or discard events. The following table lists the built-in interceptors in Flume. A complex example is used in this section. Other users can configure and use interceptions as required.

NOTE

- The interceptor is used between the sources and channels of Flume. Most sources provide parameters for configuring interceptors. You can set the parameters as required.
- Flume allows multiple interceptors to be configured for a source. The interceptor names are separated by spaces.
- The specified interceptor sequence is the order in which they are called.
- The contents inserted by the interceptor in the header can be read and used in sink.

Table 7-6 Types of built-in interceptors in Flume

Interceptor Type	Description
Timestamp Interceptor	The interceptor inserts a timestamp into the header of an event.
Host Interceptor	The interceptor inserts the IP address or host name of the node where the agent is located into the Header of an event.
Remove Header Interceptor	The interceptor discards the corresponding event based on the strings that matches the regular expression contained in the event header.

Interceptor Type	Description
UUID Interceptor	The interceptor generates a UUID string for the header of each event.
Search and Replace Interceptor	The interceptor provides a simple string-based search and replacement function based on Java regular expressions. The rule is the same as that of Java Matcher.replaceAll().
Regex Filtering Interceptor	The interceptor uses the body of an event as a text file and matches the configured regular expression to filter events. The provided regular expression can be used to exclude or include events.
Regex Extractor Interceptor	The interceptor extracts content from the original events using a regular expression and adds the content to the header of events.

Regex Filtering Interceptor is used as an example to describe how to use the interceptor. (For other types of interceptions, see the configuration provided on the official website.)

Table 7-7 Parameter configuration for **Regex Filtering Interceptor**

Parameter	Default Value	Description
type	-	Specifies the component type name. The value must be **regex_filter**.
regex	-	Specifies the regular expression used to match events.
excludeEvents	false	By default, the matched events are collected. If this parameter is set to **true**, the matched events are deleted and the unmatched events are retained.

Configuration example (**netcat tcp** is used as the source, and **logger** is used as the sink). After configuring the preceding parameters, run the **telnet** *Host name or IP address* **44444** command on the host where the Linux operating system is run, and enter a string that complies with the regular expression and another does not comply with the regular expression. The log shows that only the matched string is transmitted.

```
#define the source, channel, sink
server.sources = r1

server.channels = c1
```



```
server.sinks = k1

#config the source
server.sources.r1.type = netcat
server.sources.r1.bind = ${Host IP address}
server.sources.r1.port = 44444
server.sources.r1.interceptors= i1
server.sources.r1.interceptors.i1.type= regex_filter
server.sources.r1.interceptors.i1.regex= (flume)|(myflume)
server.sources.r1.interceptors.i1.excludeEvents= false
server.sources.r1.channels = c1

#config the channel
server.channels.c1.type = memory
server.channels.c1.capacity = 1000
server.channels.c1.transactionCapacity = 100
#config the sink
server.sinks.k1.type = logger
server.sinks.k1.channel = c1
```

NOTE

- The value of **BatchSize** of the Sink must be less than that of **transactionCapacity** of the Channel.
- Only some parameters of Source, Channel, and Sink are displayed on the Flume configuration tool page. For details, see the following configurations.
- The Customer Source, Customer Channel, and Customer Sink displayed on the Flume configuration tool page need to be configured based on self-developed code. The following common configurations are not displayed.

Common Source Configurations

- **Avro Source**

  An Avro source listens to the Avro port, receives data from the external Avro client, and places data into configured channels. Common configurations are as follows:

  Table 7-8 Common configurations of an Avro source

Parameter	Default Value	Description
channels	-	Specifies the channel connected to the source. Multiple channels can be configured.
type	avro	Specifies the type of the avro source, which must be **avro**.
bind	-	Specifies the listening host name/IP address.
port	-	Specifies the bound listening port. Ensure that this port is not occupied.

Parameter	Default Value	Description
threads	-	Specifies the maximum number of source threads.
compression-type	none	Specifies the message compression format, which can be set to **none** or **deflate**. **none** indicates that data is not compressed, while **deflate** indicates that data is compressed.
compression-level	6	Specifies the data compression level, which ranges from **1** to **9**. The larger the value is, the higher the compression rate is.
ssl	false	Specifies whether to use SSL encryption. If this parameter is set to **true**, **keystore** and **keystore-password** must be specified.
truststore-type	JKS	Specifies the Java trust store type, which can be set to **JKS** or **PKCS12**. **NOTE** Different passwords are used to protect the key store and private key of **JKS**, while the same password is used to protect the key store and private key of **PKCS12**.
truststore	-	Specifies the Java trust store file.
truststore-password	-	Specifies the Java trust store password.

Parameter	Default Value	Description
keystore-type	JKS	Specifies the keystore type set after SSL is enabled, which can be set to **JKS** or **PKCS12**. **NOTE** Different passwords are used to protect the key store and private key of **JKS**, while the same password is used to protect the key store and private key of **PKCS12**.
keystore	-	Specifies the keystore file path set after SSL is enabled. This parameter is mandatory if SSL is enabled.
keystore-password	-	Specifies the keystore password set after SSL is enabled. This parameter is mandatory if SSL is enabled.
trust-all-certs	false	Specifies whether to disable the check for the SSL server certificate. If this parameter is set to **true**, the SSL server certificate of the remote source is not checked. You are not advised to perform this operation during the production.
exclude-protocols	SSLv3	Specifies the excluded protocols. The entered protocols must be separated by spaces. The default value is **SSLv3**.
ipFilter	false	Specifies whether to enable the IP address filtering.

Parameter	Default Value	Description
ipFilter.rules	-	Specifies the rules of *N* network **ipFilters**. Host names or IP addresses must be separated by commas (,). If this parameter is set to **true**, there are two configuration rules: allow and forbidden. The configuration format is as follows: ipFilterRules=allow:ip:127.*, allow:name:localhost, deny:ip:*

- **SpoolDir Source**

  SpoolDir Source monitors and transmits new files that have been added to directories in real-time mode. Common configurations are as follows:

  Table 7-9 Common configurations of a Spooling Directory source

Parameter	Default Value	Description
channels	-	Specifies the channel connected to the source. Multiple channels can be configured.
type	spooldir	Specifies the type of the spooling source, which must be set to **spooldir**.
spoolDir	-	Specifies the monitoring directory of the Spooldir source. A Flume running user must have the read, write, and execution permissions on the directory.
monTime	0 (Disabled)	Specifies the thread monitoring threshold. When the update time exceeds the threshold, the source is restarted. Unit: second
fileSuffix	.COMPLETED	Specifies the suffix added after file transmission is complete.

Parameter	Default Value	Description
deletePolicy	never	Specifies the source file deletion policy after file transmission is complete. The value can be either **never** or **immediate**. **never** indicates that the source file is not deleted after file transmission is complete, while **immediate** indicates that the source file is immediately deleted after file transmission is complete.
ignorePattern	^$	Specifies the regular expression of a file to be ignored. The default value is ^$, indicating that spaces are ignored.
includePattern	^.*$	Specifies the regular expression that contains a file. This parameter can be used together with **ignorePattern**. If a file meets both **ignorePattern** and **includePattern**, the file is ignored. In addition, when a file starts with a period (.), the file will not be filtered.
trackerDir	.flumespool	Specifies the metadata storage path during data transmission.
batchSize	1000	Specifies the number of events written to the channel in batches.
decodeErrorPolicy	FAIL	Specifies the code error policy. **NOTE** If a code error occurs in the file, set **decodeErrorPolicy** to **REPLACE** or **IGNORE**. Flume will skip the code error and continue to collect subsequent logs.
deserializer	LINE	Specifies the file parser. The value can be either **LINE** or **BufferedLine**. • When the value is set to **LINE**, characters read from the file are transcoded one by one. • When the value is set to **BufferedLine**, one line or multiple lines of characters read from the file are transcoded in batches, which delivers better performance.
deserializer.maxLineLength	2048	Specifies the maximum length for resolution by line.

Parameter	Default Value	Description
deserializer.maxBatchLine	1	Specifies the maximum number of lines for resolution by line. If multiple lines are set, **maxLineLength** must be set to a corresponding multiplier. **NOTE** When configuring the Interceptor, take the multi-line combination into consideration to avoid data loss. If the Interceptor cannot process combined lines, set this parameter to **1**.
selector.type	replicating	Specifies the selector type. The value can be either **replicating** or **multiplexing**. **replicating** indicates that data is replicated and then transferred to each channel so that each channel receives the same data, while **multiplexing** indicates that a channel is selected based on the value of the header in the event and each channel has different data.
interceptors	-	Specifies the interceptor. Multiple interceptors are separated by spaces.
inputCharset	UTF-8	Specifies the encoding format of a read file. The encoding format must be the same as that of the data source file that has been read. Otherwise, an error may occur during character parsing.
fileHeader	false	Specifies whether to add the file name (including the file path) to the event header.
fileHeaderKey	-	Specifies that the data storage structure in header is set in the <key,value> mode. Parameters **fileHeaderKey** and **fileHeader** must be used together. Following is an example if **fileHeader** is set to true: Define **fileHeaderKey** as **file**. When the **/root/a.txt** file is read, **fileHeaderKey** exists in the header in the **file=/root/a.txt** format.
basenameHeader	false	Specifies whether to add the file name (excluding the file path) to the event header.

Parameter	Default Value	Description
basenameHeaderKey	-	Specifies that the data storage structure in header is set in the <key,value> mode. Parameters **basenameHeaderKey** and **basenameHeader** must be used together. Following is an example if **basenameHeader** is set to **true**: Define **basenameHeaderKey** as **file**. When the **a.txt** file is read, **fileHeaderKey** exists in the header in the **file=a.txt** format.
pollDelay	500	Specifies the delay for polling new files in the monitoring directory. Unit: milliseconds
recursiveDirectorySearch	false	Specifies whether to monitor new files in the subdirectory of the configured directory.
consumeOrder	oldest	Specifies the consumption order of files in a directory. If this parameter is set to **oldest** or **youngest**, the sequence of files to be read is determined by the last modification time of files in the monitored directory. If there are a large number of files in the directory, it takes a long time to search for **oldest** or **youngest** files. If this parameter is set to **random**, an earlier created file may not be read for a long time. If this parameter is set to **oldest** or **youngest**, it takes a long time to find the latest and the earliest file. The options are as follows: **random**, **youngest**, and **oldest**.
maxBackoff	4000	Specifies the maximum time to wait between consecutive attempts to write to a channel if the channel is full. If the time exceeds the threshold, an exception is thrown. The corresponding source starts to write at a smaller time value. Each time the source attempts, the digital exponent increases until the current specified value is reached. If data cannot be written, the data write fails. Unit: second
emptyFileEvent	true	Specifies whether to collect empty file information and send it to the sink end. The default value is **true**, indicating that empty file information is sent to the sink end. This parameter is valid only for HDFS Sink. Taking HDFS Sink as an example, if this parameter is set to **true** and an empty file exists in the **spoolDir** directory, an empty file with the same name will be created in the **hdfs.path** directory of HDFS.

NOTE

SpoolDir Source ignores the last line feed character of each event when data is reading by row. Therefore, Flume does not calculate the data volume counters used by the last line feed character.

- **Kafka Source**

  A Kafka source consumes data from Kafka topics. Multiple sources can consume data of the same topic, and the sources consume different partitions of the topic. Common configurations are as follows:

  Table 7-10 Common configurations of a Kafka source

Parameter	Default Value	Description
channels	-	Specifies the channel connected to the source. Multiple channels can be configured.
type	org.apache.flume.source.kafka.KafkaSource	Specifies the type of the Kafka source, which must be set to **org.apache.flume.source.kafka.KafkaSource**.
kafka.bootstrap.servers	-	Specifies the bootstrap address port list of Kafka. If Kafka has been installed in the cluster and the configuration has been synchronized to the server, you do not need to set this parameter on the server. The default value is the list of all brokers in the Kafka cluster. This parameter must be configured on the client. Use commas (,) to separate multiple values of *IP address:Port number*. The rules for matching ports and security protocols must be as follows: port 21007 matches the security mode (SASL_PLAINTEXT), and port 9092 matches the common mode (PLAINTEXT).
kafka.topics	-	Specifies the list of subscribed Kafka topics, which are separated by commas (,).
kafka.topics.regex	-	Specifies the subscribed topics that comply with regular expressions. **kafka.topics.regex** has a higher priority than **kafka.topics** and will overwrite **kafka.topics**.

Parameter	Default Value	Description
monTime	0 (Disabled)	Specifies the thread monitoring threshold. When the update time exceeds the threshold, the source is restarted. Unit: second
nodatatime	0 (Disabled)	Specifies the alarm threshold. An alarm is triggered when the duration that Kafka does not release data to subscribers exceeds the threshold. Unit: second This parameter can be configured in the **properties.properties** file.
batchSize	1000	Specifies the number of events written to the channel in batches.
batchDurationMillis	1000	Specifies the maximum duration of topic data consumption at a time, expressed in milliseconds.
keepTopicInHeader	false	Specifies whether to save topics in the event header. If the parameter value is **true**, topics configured in Kafka Sink become invalid.
setTopicHeader	true	If this parameter is set to **true**, the topic name defined in **topicHeader** is stored in the header.
topicHeader	topic	When **setTopicHeader** is set to **true**, this parameter specifies the name of the topic received by the storage device. If the property is used with that of Kafka Sink **topicHeader**, be careful not to send messages to the same topic cyclically.
useFlumeEventFormat	false	By default, an event is transferred from a Kafka topic to the body of the event in the form of bytes. If this parameter is set to **true**, the Avro binary format of Flume is used to read events. When used together with the **parseAsFlumeEvent** parameter with the same name in KafkaSink or KakfaChannel, any set **header** generated from the data source is retained.

Parameter	Default Value	Description
keepPartitionInHeader	false	Specifies whether to save partition IDs in the event header. If the parameter value is **true**, Kafka Sink writes data to the corresponding partition.
kafka.consumer.group.id	flume	Specifies the Kafka consumer group ID. Sources or proxies having the same ID are in the same consumer group.
kafka.security.protocol	SASL_PLAINTEXT	Specifies the Kafka security protocol. The parameter value must be set to PLAINTEXT in a common cluster. The rules for matching ports and security protocols must be as follows: port 21007 matches the security mode (SASL_PLAINTEXT), and port 9092 matches the common mode (PLAINTEXT).
Other Kafka Consumer Properties	-	Specifies other Kafka configurations. This parameter can be set to any consumption configuration supported by Kafka, and the **.kafka** prefix must be added to the configuration.

- **Taildir Source**

  A Taildir source monitors file changes in a directory and automatically reads the file content. In addition, it can transmit data in real time. Common configurations are as follows:

  Table 7-11 Common configurations of a Taildir source

Parameter	Default Value	Description
channels	-	Specifies the channel connected to the source. Multiple channels can be configured.
type	TAILDIR	Specifies the type of the taildir source, which must be set to TAILDIR.
filegroups	-	Specifies the group name of a collection file directory. Group names are separated by spaces.

Parameter	Default Value	Description
filegroups.<filegroupName>	-	Specifies the file path. The value must be an absolute path.
filegroups.<filegroupName>.parentDir	-	Specifies the parent directory. The value must be an absolute path.
filegroups.<filegroupName>.filePattern	-	Specifies the relative file path of the file group's parent directory. Directories can be included and regular expressions are supported. It must be used together with **parentDir**.
positionFile	-	Specifies the metadata storage path during data transmission.
headers.<filegroupName>.<headerKey>	-	Specifies the key-value of an event when data of a group is being collected.
byteOffsetHeader	false	Specifies whether each event header contains the event location information in the source file. If the parameter value is true, the location information is saved in the byteoffset variable.
maxBatchCount	Long.MAX_VALUE	Specifies the maximum number of batches that can be consecutively read from a file. If the monitored directory reads multiple files consecutively and one of the files is written at a rapid rate, other files may fail to be processed. This is because the file that is written at a high speed will be in an infinite read loop. In this case, set this parameter to a smaller value.
skipToEnd	false	Specifies whether Flume can locate the latest location of a file and read the latest data after restart. If the parameter value is true, Flume locates and reads the latest file data after restart.
idleTimeout	120000	Specifies the idle duration during file reading, expressed in milliseconds. If file content is not changed in the preset time duration, close the file. If data is written to this file after the file is closed, open the file and read data.

Parameter	Default Value	Description
writePosInterval	3000	Specifies the interval for writing metadata to a file, expressed in milliseconds.
batchSize	1000	Specifies the number of events written to the channel in batches.
monTime	0 (Disabled)	Specifies the thread monitoring threshold. When the update time exceeds the threshold, the source is restarted. Unit: second
fileHeader	false	Specifies whether to add the file name (including the file path) to the event header.
fileHeaderKey	file	Specifies that the data storage structure in header is set in the <key,value> mode. Parameters **fileHeaderKey** and **fileHeader** must be used together. Following is an example if **fileHeader** is set to true: Define **fileHeaderKey** as **file**. When the **/root/a.txt** file is read, **fileHeaderKey** exists in the header in the **file=/root/a.txt** format.

- **Http Source**

  An HTTP source receives data from an external HTTP client and sends the data to the configured channels. Common configurations are as follows:

  Table 7-12 Common configurations of an HTTP source

Parameter	Default Value	Description
channels	-	Specifies the channel connected to the source. Multiple channels can be configured.
type	http	Specifies the type of the http source, which must be set to http.
bind	-	Specifies the listening host name/IP address.
port	-	Specifies the bound listening port. Ensure that this port is not occupied.

Parameter	Default Value	Description
handler	org.apache.flume.source.http.JSONHandler	Specifies the message parsing method of an HTTP request. Two formats are supported: JSON (org.apache.flume.source.http.JSONHandler) and BLOB (org.apache.flume.sink.solr.morphline.BlobHandler).
handler.*	-	Specifies handler parameters.
exclude-protocols	SSLv3	Specifies the excluded protocols. The entered protocols must be separated by spaces. The default value is **SSLv3**.
include-cipher-suites	-	Specifies the included protocols. The entered protocols must be separated by spaces. If this parameter is left empty, all protocols are supported by default.
enableSSL	false	Specifies whether SSL is enabled in HTTP. If this parameter is set to **true**, **keystore** and **keystore-password** must be specified.
keystore-type	JKS	Specifies the keystore type, which can be **JKS** or **PKCS12**.
keystore	-	Specifies the keystore path set after SSL is enabled in HTTP.
keystorePassword	-	Specifies the keystore password set after SSL is enabled in HTTP.

- **Thrift Source**

  Thrift Source monitors the thrift port, receives data from the external Thrift clients, and puts the data into the configured channel. Common configurations are as follows:

Parameter	Default Value	Description
channels	-	Specifies the channel connected to the source. Multiple channels can be configured.
type	thrift	Specifies the type of the thrift source, which must be set to **thrift**.
bind	-	Specifies the listening host name/IP address.

Parameter	Default Value	Description
port	-	Specifies the bound listening port. Ensure that this port is not occupied.
threads	-	Specifies the maximum number of worker threads that can be run.
kerberos	false	Specifies whether Kerberos authentication is enabled.
agent-keytab	-	Specifies the address of the keytab file used by the server. The machine-machine account must be used. You are advised to use **flume/conf/flume_server.keytab** in the Flume service installation directory.
agent-principal	-	Specifies the principal of the security user used by the server. The principal must be a machine-machine account. You are advised to use the default user of Flume: flume_server/hadoop.*<system domain name>*@*<system domain name>* **NOTE** **flume_server/hadoop.**<*system domain name*> is the username. All letters in the system domain name contained in the username are lowercase letters. For example, **Local Domain** is set to **9427068F-6EFA-4833-B43E-60CB641E5B6C.COM**, and the username is **flume_server/hadoop.9427068f-6efa-4833-b43e-60cb641e5b6c.com**.
compression-type	none	Specifies the message compression format, which can be set to **none** or **deflate**. **none** indicates that data is not compressed, while **deflate** indicates that data is compressed.
ssl	false	Specifies whether to use SSL encryption. If this parameter is set to **true**, **keystore** and **keystore-password** must be specified.
keystore-type	JKS	Specifies the keystore type set after SSL is enabled.
keystore	-	Specifies the keystore file path set after SSL is enabled. This parameter is mandatory if SSL is enabled.
keystore-password	-	Specifies the keystore password set after SSL is enabled. This parameter is mandatory if SSL is enabled.

Parameter	Default Value	Description
truststore-type	JKS	Specifies the Java trust store type, which can be set to **JKS** or **PKCS12**. **NOTE** Different passwords are used to protect the key store and private key of **JKS**, while the same password is used to protect the key store and private key of **PKCS12**.
truststore	-	Specifies the Java trust store file.
truststore-password	-	Specifies the Java trust store password.

Common Channel Configurations

- **Memory Channel**

  A memory channel uses memory as the cache. Events are stored in memory queues. Common configurations are as follows:

  Table 7-13 Common configurations of a memory channel

Parameter	Default Value	Description
type	-	Specifies the type of the memory channel, which must be set to **memory**.
capacity	10000	Specifies the maximum number of events cached in a channel.
transactionCapacity	1000	Specifies the maximum number of events accessed each time. **NOTE** • The parameter value must be greater than the batchSize of the source and sink. • The value of **transactionCapacity** must be less than or equal to that of **capacity**.
channelfullcount	10	Specifies the channel full count. When the count reaches the threshold, an alarm is reported.

Parameter	Default Value	Description
keep-alive	3	Specifies the waiting time of the Put and Take threads when the transaction or channel cache is full. Unit: second
byteCapacity	80% of the maximum JVM memory	Specifies the total bytes of all event bodies in a channel. The default value is the 80% of the maximum JVM memory (indicated by **-Xmx**). Unit: bytes
byteCapacityBufferPercentage	20	Specifies the percentage of bytes in a channel (%).

- **File Channel**

  A file channel uses local disks as the cache. Events are stored in the folder specified by **dataDirs**. Common configurations are as follows:

  Table 7-14 Common configurations of a file channel

Parameter	Default Value	Description
type	-	Specifies the type of the file channel, which must be set to **file**.
checkpointDir	${BIGDATA_DATA_HOME}/hadoop/data1~N/flume/checkpoint **NOTE** This path is changed with the custom data path.	Specifies the checkpoint storage directory.
dataDirs	${BIGDATA_DATA_HOME}/hadoop/data1~N/flume/data **NOTE** This path is changed with the custom data path.	Specifies the data cache directory. Multiple directories can be configured to improve performance. The directories are separated by commas (,).
maxFileSize	2146435071	Specifies the maximum size of a single cache file, expressed in bytes.

Parameter	Default Value	Description
minimumRequiredSpace	524288000	Specifies the minimum idle space in the cache, expressed in bytes.
capacity	1000000	Specifies the maximum number of events cached in a channel.
transactionCapacity	10000	Specifies the maximum number of events accessed each time. **NOTE** • The parameter value must be greater than the batchSize of the source and sink. • The value of **transactionCapacity** must be less than or equal to that of **capacity**.
channelfullcount	10	Specifies the channel full count. When the count reaches the threshold, an alarm is reported.
useDualCheckpoints	false	Specifies the backup checkpoint. If this parameter is set to **true**, the **backupCheckpointDir** parameter value must be set.
backupCheckpointDir	-	Specifies the path of the backup checkpoint.
checkpointInterval	30000	Specifies the check interval, expressed in seconds.
keep-alive	3	Specifies the waiting time of the Put and Take threads when the transaction or channel cache is full. Unit: second
use-log-replay-v1	false	Specifies whether to enable the old reply logic.

Parameter	Default Value	Description
use-fast-replay	false	Specifies whether to enable the queue reply.
checkpointOnClose	true	Specifies that whether a checkpoint is created when a channel is disabled.

- **Memory File Channel**

  A memory file channel uses both memory and local disks as its cache and supports message persistence. It provides similar performance as a memory channel and better performance than a file channel. This channel is currently experimental and not recommended for use in production. The following table describes common configuration items: Common configurations are as follows:

  Table 7-15 Common configurations of a memory file channel

Parameter	Default Value	Description
type	org.apache.flume.channel.MemoryFileChannel	Specifies the type of the memory file channel, which must be set to **org.apache.flume.channel.MemoryFileChannel**.
capacity	50000	Specifies the maximum number of events cached in a channel.
transactionCapacity	5000	Specifies the maximum number of events processed by a transaction. **NOTE** • The parameter value must be greater than the batchSize of the source and sink. • The value of **transactionCapacity** must be less than or equal to that of **capacity**.

Parameter	Default Value	Description
subqueueByteCapacity	20971520	Specifies the maximum size of events that can be stored in a subqueue, expressed in bytes. A memory file channel uses both queues and subqueues to cache data. Events are stored in a subqueue, and subqueues are stored in a queue. **subqueueCapacity** and **subqueueInterval** determine the size of events that can be stored in a subqueue. **subqueueCapacity** specifies the capacity of a subqueue, and **subqueueInterval** specifies the duration that a subqueue can store events. Events in a subqueue are sent to the destination only after the subqueue reaches the upper limit of **subqueueCapacity** or **subqueueInterval**. **NOTE** The value of **subqueueByteCapacity** must be greater than the number of events specified by **batchSize**.
subqueueInterval	2000	Specifies the maximum duration that a subqueue can store events, expressed in milliseconds.
keep-alive	3	Specifies the waiting time of the Put and Take threads when the transaction or channel cache is full. Unit: second
dataDir	-	Specifies the cache directory for local files.
byteCapacity	80% of the maximum JVM memory	Specifies the channel cache capacity. Unit: bytes
compression-type	None	Specifies the message compression format, which can be set to **none** or **deflate**. **none** indicates that data is not compressed, while **deflate** indicates that data is compressed.
channelfullcount	10	Specifies the channel full count. When the count reaches the threshold, an alarm is reported.

The following is a configuration example of a memory file channel:


```
server.channels.c1.type = org.apache.flume.channel.MemoryFileChannel
server.channels.c1.dataDir = /opt/flume/mfdata
server.channels.c1.subqueueByteCapacity = 20971520
server.channels.c1.subqueueInterval=2000
server.channels.c1.capacity = 500000
server.channels.c1.transactionCapacity = 40000
```

- **Kafka Channel**

  A Kafka channel uses a Kafka cluster as the cache. Kafka provides high availability and multiple copies to prevent data from being immediately consumed by sinks when Flume or Kafka Broker crashes.

  Table 7-16 Common configurations of a Kafka channel

Parameter	Default Value	Description
type	-	Specifies the type of the Kafka channel, which must be set to **org.apache.flume.channel.kafka.KafkaChannel**.
kafka.bootstrap.servers	-	Specifies the bootstrap address port list of Kafka. If Kafka has been installed in the cluster and the configuration has been synchronized to the server, you do not need to set this parameter on the server. The default value is the list of all brokers in the Kafka cluster. This parameter must be configured on the client. Use commas (,) to separate multiple values of *IP address:Port number*. The rules for matching ports and security protocols must be as follows: port 21007 matches the security mode (SASL_PLAINTEXT), and port 9092 matches the common mode (PLAINTEXT).

Parameter	Default Value	Description
kafka.topic	flume-channel	Specifies the Kafka topic used by the channel to cache data.
kafka.consumer.group.id	flume	Specifies the data group ID obtained from Kafka. This parameter cannot be left blank.
parseAsFlumeEvent	true	Specifies whether data is parsed into Flume events.
migrateZookeeperOffsets	true	Specifies whether to search for offsets in ZooKeeper and submit them to Kafka when there is no offset in Kafka.
kafka.consumer.auto.offset.reset	latest	Specifies where to consume if there is no offset record, which can be set to **earliest**, **latest**, or **none**. **earliest** indicates that the offset is reset to the initial point, **latest** indicates that the offset is set to the latest position, and **none** indicates that an exception is thrown if there is no offset.

Parameter	Default Value	Description
kafka.producer.security.protocol	SASL_PLAINTEXT	Specifies the Kafka producer security protocol. The rules for matching ports and security protocols must be as follows: port 21007 matches the security mode (SASL_PLAINTEXT), and port 9092 matches the common mode (PLAINTEXT). **NOTE** If the parameter is not displayed, click **+** in the lower left corner of the dialog box to display all parameters.
kafka.consumer.security.protocol	SASL_PLAINTEXT	Specifies the Kafka consumer security protocol. The rules for matching ports and security protocols must be as follows: port 21007 matches the security mode (SASL_PLAINTEXT), and port 9092 matches the common mode (PLAINTEXT).
pollTimeout	500	Specifies the maximum timeout interval for the consumer to invoke the poll function. Unit: milliseconds
ignoreLongMessage	false	Specifies whether to discard oversized messages.
messageMaxLength	1000012	Specifies the maximum length of a message written by Flume to Kafka.

Common Sink Configurations

- **HDFS Sink**

  An HDFS sink writes data into HDFS. Common configurations are as follows:

Table 7-17 Common configurations of an HDFS sink

Parameter	Default Value	Description
channel	-	Specifies the channel connected to the sink.
type	hdfs	Specifies the type of the hdfs sink, which must be set to **hdfs**.
hdfs.path	-	Specifies the data storage path in HDFS. The value must start with **hdfs://hacluster/**.
monTime	0 (Disabled)	Specifies the thread monitoring threshold. When the update time exceeds the threshold, the sink is restarted. Unit: second
hdfs.inUseSuffix	.tmp	Specifies the suffix of the HDFS file to which data is being written.
hdfs.rollInterval	30	Specifies the interval for file rolling, expressed in seconds. Set **hdfs.fileCloseByEndEvent** to **false** if you set this parameter.
hdfs.rollSize	1024	Specifies the size for file rolling, expressed in bytes. Set **hdfs.fileCloseByEndEvent** to **false** if you set this parameter.
hdfs.rollCount	10	Specifies the number of events for file rolling. Set **hdfs.fileCloseByEndEvent** to **false** if you set this parameter. **NOTE** Parameters **rollInterval**, **rollSize**, and **rollCount** can be configured at the same time. The parameter meeting the requirements takes precedence for compression.
hdfs.idleTimeout	0	Specifies the timeout interval for closing idle files automatically, expressed in seconds.
hdfs.batchSize	1000	Specifies the number of events written into HDFS in batches.
hdfs.kerberosPrincipal	-	Specifies the Kerberos principal of HDFS authentication. This parameter is mandatory in a secure mode, but not required in a common mode.

Parameter	Default Value	Description
hdfs.kerberosKeytab	-	Specifies the Kerberos keytab of HDFS authentication. This parameter is not required in a common mode, but in a secure mode, the Flume running user must have the permission to access **keyTab** path in the **jaas.cof** file.
hdfs.fileCloseByEndEvent	true	Specifies whether to close the HDFS file when the last event of the source file is received.
hdfs.batchCallTimeout	-	Specifies the timeout control duration when events are written into HDFS in batches. Unit: milliseconds If this parameter is not specified, the timeout duration is controlled when each event is written into HDFS. When the value of **hdfs.batchSize** is greater than 0, configure this parameter to improve the performance of writing data into HDFS. **NOTE** The value of **hdfs.batchCallTimeout** depends on **hdfs.batchSize**. A greater **hdfs.batchSize** requires a larger **hdfs.batchCallTimeout**. If the value of **hdfs.batchCallTimeout** is too small, writing events to HDFS may fail.
serializer.appendNewline	true	Specifies whether to add a line feed character (**\n**) after an event is written to HDFS. If a line feed character is added, the data volume counters used by the line feed character will not be calculated by HDFS sinks.
hdfs.filePrefix	over_%{basename}	Specifies the file name prefix after data is written to HDFS.
hdfs.fileSuffix	-	Specifies the file name suffix after data is written to HDFS.
hdfs.inUsePrefix	-	Specifies the prefix of the HDFS file to which data is being written.

Parameter	Default Value	Description
hdfs.fileType	DataStream	Specifies the HDFS file format, which can be set to **SequenceFile**, **DataStream**, or **CompressedStream**. **NOTE** If the parameter is set to **SequenceFile** or **DataStream**, output files are not compressed, and the **codeC** parameter cannot be configured. However, if the parameter is set to **CompressedStream**, the output files are compressed, and the **codeC** parameter must be configured together.
hdfs.codeC	-	Specifies the file compression format, which can be set to **gzip**, **bzip2**, **lzo**, **lzop**, or **snappy**.
hdfs.maxOpenFiles	5000	Specifies the maximum number of HDFS files that can be opened. If the number of opened files reaches this value, the earliest opened files are closed.
hdfs.writeFormat	Writable	Specifies the file write format, which can be set to **Writable** or **Text**.
hdfs.callTimeout	10000	Specifies the timeout control duration each time events are written into HDFS, expressed in milliseconds.
hdfs.threadsPoolSize	-	Specifies the number of threads used by each HDFS sink for HDFS I/O operations.
hdfs.rollTimerPoolSize	-	Specifies the number of threads used by each HDFS sink to schedule the scheduled file rolling.
hdfs.round	false	Specifies whether to round off the timestamp value. If this parameter is set to true, all time-based escape sequences (except %t) are affected.
hdfs.roundUnit	second	Specifies the unit of the timestamp value that has been rounded off, which can be set to **second**, **minute**, or **hour**.
hdfs.useLocalTimeStamp	true	Specifies whether to enable the local timestamp. The recommended parameter value is **true**.

Parameter	Default Value	Description
hdfs.closeTries	0	Specifies the maximum attempts for the **hdfs sink** to stop renaming a file. If the parameter is set to the default value **0**, the sink does not stop renaming the file until the file is successfully renamed.
hdfs.retryInterval	180	Specifies the interval of request for closing the HDFS file, expressed in seconds. **NOTE** For each closing request, there are multiple RPCs working on the NameNode back and forth, which may make the NameNode overloaded if the parameter value is too small. Also, when the parameter is set to **0**, the Sink will not attempt to close the file, but opens the file or uses **.tmp** as the file name extension, if the first closing attempt fails.
hdfs.failcount	10	Specifies the number of times that data fails to be written to HDFS. If the number of times that the sink fails to write data to HDFS exceeds the parameter value, an alarm indicating abnormal data transmission is reported.

- **Avro Sink**

  An Avro sink converts events into Avro events and sends them to the monitoring ports of the hosts. Common configurations are as follows:

  Table 7-18 Common configurations of an Avro sink

Parameter	Default Value	Description
channel	-	Specifies the channel connected to the sink.
type	-	Specifies the type of the avro sink, which must be set to **avro**.
hostname	-	Specifies the bound host name or IP address.
port	-	Specifies the bound listening port. Ensure that this port is not occupied.

Parameter	Default Value	Description
batch-size	1000	Specifies the number of events sent in a batch.
client.type	DEFAULT	Specifies the client instance type. Set this parameter based on the communication protocol used by the configured model. The options are as follows: • **DEFAULT**: The client instance of the AvroRPC type is returned. • **OTHER**: NULL is returned. • **THRIFT**: The client instance of the Thrift RPC type is returned. • **DEFAULT_LOADBALANCING**: The client instance of the LoadBalancing RPC type is returned. • **DEFAULT_FAILOVER**: The client instance of the Failover RPC type is returned.
ssl	false	Specifies whether to use SSL encryption. If this parameter is set to **true**, **keystore** and **keystore-password** must be specified.
truststore-type	JKS	Specifies the Java trust store type, which can be set to **JKS** or **PKCS12**. **NOTE** Different passwords are used to protect the key store and private key of **JKS**, while the same password is used to protect the key store and private key of **PKCS12**.

Parameter	Default Value	Description
truststore	-	Specifies the Java trust store file.
truststore-password	-	Specifies the Java trust store password.
keystore-type	JKS	Specifies the keystore type set after SSL is enabled.
keystore	-	Specifies the keystore file path set after SSL is enabled. This parameter is mandatory if SSL is enabled.
keystore-password	-	Specifies the keystore password after SSL is enabled. This parameter is mandatory if SSL is enabled.
connect-timeout	20000	Specifies the timeout for the first connection, expressed in milliseconds.
request-timeout	20000	Specifies the maximum timeout for a request after the first request, expressed in milliseconds.
reset-connection-interval	0	Specifies the interval between a connection failure and a second connection, expressed in seconds. If the parameter is set to **0**, the system continuously attempts to perform a connection.

Parameter	Default Value	Description
compression-type	none	Specifies the compression type of the batch data, which can be set to **none** or **deflate**. **none** indicates that data is not compressed, while **deflate** indicates that data is compressed. This parameter value must be the same as that of the AvroSource compression-type.
compression-level	6	Specifies the compression level of batch data, which can be set to **1** to **9**. A larger value indicates a higher compression rate.
exclude-protocols	SSLv3	Specifies the excluded protocols. The entered protocols must be separated by spaces. The default value is **SSLv3**.

- **HBase Sink**

  An HBase sink writes data into HBase. Common configurations are as follows:

  Table 7-19 Common configurations of an HBase sink

Parameter	Default Value	Description
channel	-	Specifies the channel connected to the sink.
type	-	Specifies the type of the HBase sink, which must be set to **hbase**.
table	-	Specifies the HBase table name.
columnFamily	-	Specifies the HBase column family.
monTime	0 (Disabled)	Specifies the thread monitoring threshold. When the update time exceeds the threshold, the sink is restarted. Unit: second

Parameter	Default Value	Description
batchSize	1000	Specifies the number of events written into HBase in batches.
kerberosPrincipal	-	Specifies the Kerberos principal of HBase authentication. This parameter is mandatory in a secure mode, but not required in a common mode.
kerberosKeytab	-	Specifies the Kerberos keytab of HBase authentication. This parameter is not required in a common mode, but in a secure mode, the Flume running user must have the permission to access **keyTab** path in the **jaas.cof** file.
coalesceIncrements	true	Specifies whether to perform multiple operations on the same hbase cell in a same processing batch. Setting this parameter to **true** improves performance.

- **Kafka Sink**

  A Kafka sink writes data into Kafka. Common configurations are as follows:

  Table 7-20 Common configurations of a Kafka sink

Parameter	Default Value	Description
channel	-	Specifies the channel connected to the sink.
type	-	Specifies the type of the kafka sink, which must be set to **org.apache.flume.sink.kafka.KafkaSink**.

Parameter	Default Value	Description
kafka.bootstrap.servers	-	Specifies the bootstrap address port list of Kafka. If Kafka has been installed in the cluster and the configuration has been synchronized to the server, you do not need to set this parameter on the server. The default value is the list of all brokers in the Kafka cluster. The client must be configured with this parameter. If there are multiple values, use commas (,) to separate the values. The rules for matching ports and security protocols must be as follows: port 21007 matches the security mode (SASL_PLAINTEXT), and port 9092 matches the common mode (PLAINTEXT).
monTime	0 (Disabled)	Specifies the thread monitoring threshold. When the update time exceeds the threshold, the sink is restarted. Unit: second
kafka.producer.acks	1	Successful write is determined by the number of received acknowledgement messages about replicas. The value **0** indicates that no confirm message needs to be received, the value **1** indicates that the system is only waiting for only the acknowledgement information from a leader, and the value **-1** indicates that the system is waiting for the acknowledgement messages of all replicas. If this parameter is set to **-1**, data loss can be avoided in some leader failure scenarios.
kafka.topic	-	Specifies the topic to which data is written. This parameter is mandatory.
allowTopicOverride	false	Specifies whether to replace the topic configured in **kafka.topic** with the topic saved in Event Header.
flumeBatchSize	1000	Specifies the number of events written into Kafka in batches.

Parameter	Default Value	Description
kafka.security.protocol	SASL_PLAINTEXT	Specifies the Kafka security protocol. The parameter value must be set to PLAINTEXT in a common cluster. The rules for matching ports and security protocols must be as follows: port 21007 matches the security mode (SASL_PLAINTEXT), and port 9092 matches the common mode (PLAINTEXT).
ignoreLongMessage	false	Specifies whether to discard oversized messages.
messageMaxLength	1000012	Specifies the maximum length of a message written by Flume to Kafka.
defaultPartitionId	-	Specifies the ID of the Kafka partition to which the events of a channel are transferred. The **partitionIdHeader** value overwrites this parameter value. By default, if this parameter is left blank, events will be distributed by the Kafka Producer's partitioner (by a specified key or a partitioner customized by **kafka.partitioner.class**).
partitionIdHeader	-	When you set this parameter, the sink will take the value of the field named using the value of this property from the event header and send the message to the specified partition of the topic. If the value does not have a valid partition, **EventDeliveryException** is thrown. If the header value already exists, this setting overwrites the **defaultPartitionId** parameter.
Other Kafka Producer Properties	-	Specifies other Kafka configurations. This parameter can be set to any production configuration supported by Kafka, and the **.kafka** prefix must be added to the configuration.

- **Thrift Sink**

  A Thrift sink converts events to Thrift events and sends them to the monitoring port of the configured host. Common configurations are as follows:

Table 7-21 Common configurations of a Thrift sink

Parameter	Default Value	Description
channel	-	Specifies the channel connected to the sink.
type	thrift	Specifies the type of the thrift sink, which must be set to **thrift**.
hostname	-	Specifies the bound host name or IP address.
port	-	Specifies the bound listening port. Ensure that this port is not occupied.
batch-size	1000	Specifies the number of events sent in a batch.
connect-timeout	20000	Specifies the timeout for the first connection, expressed in milliseconds.
request-timeout	20000	Specifies the maximum timeout for a request after the first request, expressed in milliseconds.
kerberos	false	Specifies whether Kerberos authentication is enabled.
client-keytab	-	Specifies the path of the client **keytab** file. The Flume running user must have the access permission on the authentication file.
client-principal	-	Specifies the principal of the security user used by the client.
server-principal	-	Specifies the principal of the security user used by the server.

Parameter	Default Value	Description
compression-type	none	Specifies the compression type of data sent by Flume, which can be set to **none** or **deflate**. **none** indicates that data is not compressed, while **deflate** indicates that data is compressed.
maxConnections	5	Specifies the maximum size of the connection pool for Flume to send data.
ssl	false	Specifies whether to use SSL encryption.
truststore-type	JKS	Specifies the Java trust store type.
truststore	-	Specifies the Java trust store file.
truststore-password	-	Specifies the Java trust store password.
reset-connection-interval	0	Specifies the interval between a connection failure and a second connection, expressed in seconds. If the parameter is set to **0**, the system continuously attempts to perform a connection.

Precautions

- What are the reliability measures of Flume?
  - Use the transaction mechanisms between Source and Channel as well as between Channel and Sink.
  - Sink Processor supports failover and load balancing. The following is an example of load balancing:

```
server.sinkgroups=g1
server.sinkgroups.g1.sinks=k1 k2
server.sinkgroups.g1.processor.type=load_balance
server.sinkgroups.g1.processor.backoff=true
server.sinkgroups.g1.processor.selector=random
```

- What are the precautions for the aggregation and cascading of multiple Flume agents?

- Avro or Thrift protocol can be used for cascading.
- When the aggregation end contains multiple nodes, evenly distribute the agents and do not aggregate all agents on a single node.

7.3 Installing the Flume Client

To use Flume to collect logs, you must install the Flume client on a log host. You can create an ECS and install the Flume client on it.

Prerequisites

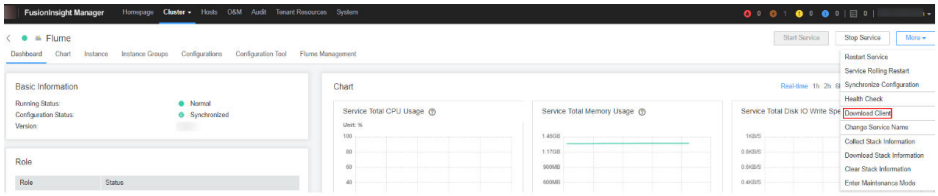
- A cluster with the Flume component has been created.
- The log host is in the same VPC and subnet with the MRS cluster.
- You have obtained the username and password for logging in to the log host.
- The installation directory is automatically created if it does not exist. If it exists, the directory must be left blank. The directory path cannot contain any space.

Installing the Flume Client

Step 1 Obtain the software package.

Log in to the FusionInsight Manager. Choose **Cluster** > *Name of the target cluster* > **Services** > **Flume**. On the Flume service page that is displayed, choose **More** > **Download Client** in the upper right corner and set **Select Client Type** to **Complete Client** to download the Flume service client file.

The file name of the client is **FusionInsight_Cluster_**<*Cluster ID*>**_Flume_Client.tar**. This section takes the client file **FusionInsight_Cluster_1_Flume_Client.tar** as an example.



Step 2 Upload the software package.

Upload the software package to a directory, for example, **/opt/client**, on the node where the Flume client is to be installed as user **user**.

NOTE

user is the user who installs and runs the Flume client.

Step 3 Decompress the software package.

Log in to the node where the Flume service client is to be installed as user **user**. Go to the directory where the installation package is installed, for example, **/opt/client**, and run the following command to decompress the installation package to the current directory:

cd /opt/client

tar -xvf FusionInsight_Cluster_1_Flume_Client.tar

Step 4 Verify the software package.

Run the **sha256sum -c** command to verify the decompressed file. If **OK** is returned, the verification is successful. Example:

sha256sum -c FusionInsight_Cluster_1_Flume_ClientConfig.tar.sha256

```
FusionInsight_Cluster_1_Flume_ClientConfig.tar: OK
```

Step 5 Decompress the package.

tar -xvf FusionInsight_Cluster_1_Flume_ClientConfig.tar

Step 6 To install the Flume client on a node outside the cluster, perform the following steps to configure the installation environment. Skip this step if you install Flume client on a node in the cluster.

1. Run the following command to install the client running environment to a new directory, for example, **/opt/Flumeenv**. A directory is automatically generated during the client installation.

   sh /opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/install.sh /opt/Flumeenv

   If the following information is displayed, the client running environment is successfully installed:

   ```
   Components client installation is complete.
   ```

2. Run the following command to set environment variables:

   source /opt/Flumeenv/bigdata_env

Step 7 Check whether the number of clients is **1**.

- If yes, use the independent installation mode and go to **Step 8**. The installation is complete.
- If no, use the batch installation mode and go to **Step 9**.

Step 8 Run the following command in the Flume client installation directory to install the client to a specified directory (for example, **opt/FlumeClient**): After the client is installed successfully, the installation is complete.

cd /opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient

./install.sh -d *\/opt/FlumeClient* **-f** *MonitorServerService IP address or host name of the role* **-c** *User service configuration filePath for storing properties.properties* **-s** *CPU threshold* **-l /var/log/Bigdata -e** *FlumeServer service IP address or host name* **-n** *Flume*

NOTE

- **-d**: Flume client installation path
- (Optional) **-f**: IP addresses or host names of two MonitorServer roles. The IP addresses or host names are separated by commas (,). If this parameter is not configured, the Flume client does not send alarm information to MonitorServer and information about the client cannot be viewed on the FusionInsight Manager GUI.
- (Optional) **-c**: Service configuration file, which needs to be generated on the configuration tool page of the Flume server based on your service requirements by referring to **Flume Service Configuration Guide**. Upload the file to any directory on the node where the client is to be installed. If this parameter is not specified during the installation, you can upload the generated service configuration file **properties.properties** to the **/opt/FlumeClient/fusioninsight-flume-1.9.0/conf** directory after the installation.
- (Optional) **-s**: cgroup threshold. The value is an integer ranging from 1 to 100 x *N*. *N* indicates the number of CPU cores. The default threshold is **-1**, indicating that the processes added to the cgroup are not restricted by the CPU usage.
- (Optional) **-l**: Log path. The default value is **/var/log/Bigdata**. The user **user** must have the write permission on the directory. When the client is installed for the first time, a subdirectory named **flume-client** is generated. After the installation, subdirectories named **flume-client-***n* will be generated in sequence. The letter *n* indicates a sequence number, which starts from 1 in ascending order. In the **/conf/** directory of the Flume client installation directory, modify the **ENV_VARS** file and search for the **FLUME_LOG_DIR** attribute to view the client log path.
- (Optional) **-e**: Service IP address or host name of FlumeServer, which is used to receive statistics for the monitoring indicator reported by the client.
- (Optional) **-n**: Name of the Flume client. You can choose **Cluster** > *Name of the desired cluster* > **Service** > **Flume** > **Flume Management** on FusionInsight Manager to view the client name on the corresponding node.
- If the following error message is displayed, run the **export JAVA_HOME=*JDK path*** command. You can run the **echo $JAVA_HOME** command to query the JDK path.
  JAVA_HOME is null in current user,please install the JDK and set the JAVA_HOME
- When installing a cross-platform client in a cluster, go to the **/opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FusionInsight-Flume-1.9.0.tar.gz** directory to install the Flume client.

Step 9 Go to the directory for installing clients in batches.

cd /opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient/batch_install

NOTE

When installing a cross-platform client in a cluster, go to the **/opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FusionInsight-Flume-1.9.0.tar.gz** directory to install the Flume client.

Step 10 Configure the **host_info.cfg** file. The format of the configuration file is as follows:

host_ip="",user="",password="",install_path="",flume_config_file="",monitor_server_ip="",log_path="",flume_server_ip="",cgroup_threshold="",client_name=""

NOTE

- (Mandatory) *host_ip*: IP address of the node where the Flume client is to be installed.
- (Mandatory) *user*: User name for logging in to the node where the Flume client is to be installed remotely. The user must have the crontab access permission. You can use **echo** *Username* **> /etc/cron.allow** to grant the user the permission.
- (Mandatory) *password*: Password for logging in to the Flume client to be installed remotely. There can be security risks if a configuration file contains the authentication password. You are advised to delete the configuration file or use other secure methods to keep the password.
- (Mandatory) **install_path**: Installation path of the Flume client.
- (Optional) **flume_config_file**: Configuration file for Flume running. You are advised to specify this configuration file during Flume installation. If you do not set this parameter, retain the value "" and do not delete the parameter.
- (Optional) **monitor_server_ip**: Service IP address of the Flume MonitorServer in the cluster. You can check the IP address on FusionInsight Manager. You can select either of the two IP addresses. If the IP address is not configured, the client does not send alarm information to the cluster in the scenario where a process is faulty.
- (Optional) **log_path**: Path for storing Flume run logs. If this parameter is not set, logs are recorded in **/var/log/Bigdata/flume-client**-*Index* by default. Index value: If there is only one client in this path, the value is 1. If there are multiple clients, the index value is incremented by 1.
- (Optional) **flume_server_ip**: Service IP address of the Flume server. The indicator information of the client is reported to the cluster from this node. The indicator information about the client can be displayed on the web client. If the indicator information is not configured, the client does not display the indicator information.
- (Optional) **cgroup_threshold**: cgroup threshold. The value is an integer ranging from 1 to 100 x *N*. *N* indicates the number of CPU cores. The default threshold is **-1**, indicating that the processes added to the cgroup are not restricted by the CPU usage.
- (Optional) **client_name**: Client name. The client name is displayed on the client monitoring page. If the client name is not configured, the client name is empty.
- If multiple nodes need to be added, the format is as follows:

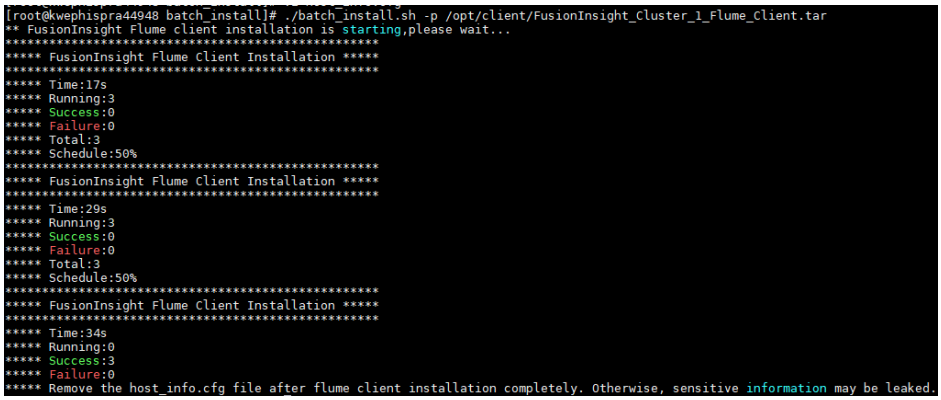
  host_ip="ip1",user="user1",password="******",install_path="/tmp/flumeclient",flume_config_file="",monitor_server_ip="xxx.xxx.xxx.xxx",log_path="",flume_server_ip="xxx.xxx.xxx.xxx",cgroup_threshold="",client_name="ip1-client-1"

  host_ip="ip2",user="user1",password="******",install_path="/tmp/flumeclient",flume_config_file="",monitor_server_ip="xxx.xxx.xxx.xxx",log_path="",flume_server_ip="xxx.xxx.xxx.xxx",cgroup_threshold="",client_name="ip2-client-1"

  host_ip="ip3",user="user1",password="******",install_path="/tmp/flumeclient",flume_config_file="",monitor_server_ip="xxx.xxx.xxx.xxx",log_path="",flume_server_ip="xxx.xxx.xxx.xxx",cgroup_threshold="",client_name="ip3-client-1"

Step 11 Run the following command to install the Flume client in batches. The installation is complete, as shown in the following figure.

./batch_install.sh -p /opt/client/FusionInsight_Cluster_1_Flume_Client.tar



Step 12 Delete the password information from the **host_info.cfg** file.

After the batch installation is complete, delete the password information from the **host_info.cfg** file immediately. Otherwise, the password may be disclosed.

----End

Flume Client Cgroup Usage Guide

This section describes how to join and log out of a cgroup, query the cgroup status, and change the cgroup CPU threshold.

- **Join Cgroup**

  Assume that the Flume client installation path is **/opt/FlumeClient**, and the cgroup CPU threshold is 50%. Run the following command to join a cgroup:

  cd /opt/FlumeClient/fusioninsight-flume-1.9.0/bin

  ./flume-manage.sh cgroup join 50

  NOTE

  - This command can be used to join a cgroup and change the cgroup CPU threshold.
  - The value of the CPU threshold of a cgroup ranges from 1 to 100 x *N*. *N* indicates the number of CPU cores.

- **Check Cgroup status**

  Assume that the Flume client installation path is **/opt/FlumeClient**. Run the following commands to query the cgroup status:

  cd /opt/FlumeClient/fusioninsight-flume-1.9.0/bin

  ./flume-manage.sh cgroup status

- **Exit Cgroup**

  Assume that the Flume client installation path is **/opt/FlumeClient**. Run the following commands to exit cgroup:

  cd /opt/FlumeClient/fusioninsight-flume-1.9.0/bin

  ./flume-manage.sh cgroup exit

NOTE

- After the client is installed, the default cgroup is automatically created. If the **-s** parameter is not configured during client installation, the default value **-1** is used. The default value indicates that the agent process is not restricted by the CPU usage.
- Joining or exiting a cgroup does not affect the agent process. Even if the agent process is not started, the joining or exiting operation can be performed successfully, and the operation will take effect after the next startup of the agent process.
- After the client is uninstalled, the cgroups created during the client installation are automatically deleted.

7.4 Quickly Using Flume to Collect Node Logs

You can use Flume to import collected log information to Kafka.

Prerequisites

- A streaming cluster that contains components such as Flume and Kafka and has Kerberos authentication enabled has been created. For details, see **Buying a Custom Cluster**.
- The streaming cluster can properly communicate with the node where logs are generated.

Using the Flume Client

NOTE

You do not need to perform **Step 2** to **Step 6** for a normal cluster.

Step 1 Install the Flume client.

Install the Flume client in a directory, for example, **/opt/Flumeclient**, on the node where logs are generated by referring to **Installing the Flume Client**. The Flume client installation directories in the following steps are only examples. Change them to the actual installation directories.

Step 2 Copy the configuration file of the authentication server from the Master1 node to the *Flume client installation directory*/**fusioninsight-flume-***Flume component version number*/**conf** directory on the node where the Flume client is installed.

The full file path is **${BIGDATA_HOME}/FusionInsight_BASE_***XXX*/**1_***X*_KerberosClient/etc/kdc.conf**. In the preceding path, **XXX** indicates the product version number. **X** indicates a random number. Replace them based on site requirements. The file must be saved by the user who installs the Flume client, for example, user **root**.

Step 3 Check the service IP address of any node where the Flume role is deployed.

Log in to FusionInsight Manager, click **Cluster**, click **Services**, and click Flume. On the displayed page, click **Instance**. For details, see **Accessing FusionInsight Manager**. Check the service IP address of any node where the Flume role is deployed.

NOTE

If the **Components** tab is unavailable, complete IAM user synchronization first. (On the **Dashboard** page, click **Synchronize** on the right side of **IAM User Sync** to synchronize IAM users.)

Step 4 Copy the user authentication file from this node to the *Flume client installation directory*/**fusioninsight-flume-***Flume component version number*/**conf** directory on the Flume client node.

The full file path is **${BIGDATA_HOME}/FusionInsight_Porter_***XXX*/**install/FusionInsight-Flume-***Flume component version number*/**flume/conf/flume.keytab**.

In the preceding paths, **XXX** indicates the product version number. Change it based on the site requirements. The file must be saved by the user who installs the Flume client, for example, user **root**.

Step 5 Copy the **jaas.conf** file from this node to the **conf** directory on the Flume client node.

The full file path is **${BIGDATA_HOME}/FusionInsight_Current/1_***X*_**Flume/etc/jaas.conf**.

In the preceding path, **X** indicates a random number. Change it based on the site requirements. The file must be saved by the user who installs the Flume client, for example, user **root**.

Step 6 Log in to the Flume client node and go to the client installation directory. Run the following command to modify the file:

vi conf/jaas.conf

Change the full path of the user authentication file defined by **keyTab** to the *Flume client installation directory*/**fusioninsight-flume-***Flume component version number*/**conf** saved in **Step 4**, and save the modification and exit.

Step 7 Run the following command to modify the **flume-env.sh** configuration file of the Flume client:

vi *Flume client installation directory*/**fusioninsight-flume-***Flume component version number*/**conf/flume-env.sh**

Add the following information after **-XX:+UseCMSCompactAtFullCollection**:

```
-Djava.security.krb5.conf=Flume client installation directory/fusioninsight-flume-1.9.0/conf/kdc.conf -Djava.security.auth.login.config=Flume client installation directory/fusioninsight-flume-1.9.0/conf/jaas.conf -Dzookeeper.request.timeout=120000
```

Example: **"-XX:+UseCMSCompactAtFullCollection -Djava.security.krb5.conf=/opt/FlumeClient/fusioninsight-flume-***Flume component version number*/**conf/kdc.conf -Djava.security.auth.login.config=/opt/FlumeClient/fusioninsight-flume-***Flume component version number*/**conf/jaas.conf -Dzookeeper.request.timeout=120000"**

Change *Flume client installation directory* to the actual installation directory. Then save and exit.

Step 8 Run the following command to restart the Flume client:

cd *Flume client installation directory*/**fusioninsight-flume-**_Flume component version number_/**bin**

./flume-manage.sh restart

Example:

cd /opt/FlumeClient/fusioninsight-flume-_Flume component version number_/**bin**

./flume-manage.sh restart

NOTE

The Flume client will be automatically restarted after being stopped. If you do not need automatic restart, run the following command:

./flume-manage.sh stop force

If you want to restart the Flume client, run the following command:

./flume-manage.sh start force

Step 9 Configure jobs based on actual service scenarios.

- Some parameters can be configured on Manager. For details, see **Non-Encrypted Transmission** or **Encrypted Transmission**.
- Set the parameters in the **properties.properties** file. The following uses SpoolDir Source+File Channel+Kafka Sink as an example.

  Run the following command on the node where the Flume client is installed. Configure and save jobs in the Flume client configuration file **properties.properties** based on actual service requirements. For details, see **Flume Service Configuration Guide**.

  vi *Flume client installation directory*/**fusioninsight-flume-**_Flume component version number_/**conf/properties.properties**

```
################################################################################
###################
client.sources = static_log_source
client.channels = static_log_channel
client.sinks = kafka_sink
################################################################################
###################
#LOG_TO_HDFS_ONLINE_1

client.sources.static_log_source.type = spooldir
client.sources.static_log_source.spoolDir = Monitoring directory
client.sources.static_log_source.fileSuffix = .COMPLETED
client.sources.static_log_source.ignorePattern = ^$
client.sources.static_log_source.trackerDir = Metadata storage path during transmission
client.sources.static_log_source.maxBlobLength = 16384
client.sources.static_log_source.batchSize = 51200
client.sources.static_log_source.inputCharset = UTF-8
client.sources.static_log_source.deserializer = LINE
client.sources.static_log_source.selector.type = replicating
client.sources.static_log_source.fileHeaderKey = file
client.sources.static_log_source.fileHeader = false
client.sources.static_log_source.basenameHeader = true
client.sources.static_log_source.basenameHeaderKey = basename
client.sources.static_log_source.deletePolicy = never

client.channels.static_log_channel.type = file
client.channels.static_log_channel.dataDirs = Data cache path. Multiple paths, separated by commas (,), can be configured to improve performance.
client.channels.static_log_channel.checkpointDir = Checkpoint storage path
client.channels.static_log_channel.maxFileSize = 2146435071
client.channels.static_log_channel.capacity = 1000000
```



```
client.channels.static_log_channel.transactionCapacity = 612000
client.channels.static_log_channel.minimumRequiredSpace = 524288000

client.sinks.kafka_sink.type = org.apache.flume.sink.kafka.KafkaSink
client.sinks.kafka_sink.kafka.topic = Topic to which data is written, for example, flume_test
client.sinks.kafka_sink.kafka.bootstrap.servers = XXX.XXX.XXX.XXX:Kafka port number,XXX.XXX.XXX.XXX:Kafka port number,XXX.XXX.XXX.XXX:Kafka port number
client.sinks.kafka_sink.flumeBatchSize = 1000
client.sinks.kafka_sink.kafka.producer.type = sync
client.sinks.kafka_sink.kafka.security.protocol = SASL_PLAINTEXT
client.sinks.kafka_sink.kafka.kerberos.domain.name = Kafka domain name. This parameter is mandatory for a security cluster, for example, hadoop.example.com.
client.sinks.kafka_sink.requiredAcks = 0

client.sources.static_log_source.channels = static_log_channel
client.sinks.kafka_sink.channel = static_log_channel
```

NOTE

- **client.sinks.kafka_sink.kafka.topic**: Topic to which data is written. If the topic does not exist in Kafka, it is automatically created by default.
- **client.sinks.kafka_sink.kafka.bootstrap.servers**: List of Kafka Brokers, which are separated by commas (,). By default, the port is **21007** for a security cluster and **9092** for a normal cluster.
- **client.sinks.kafka_sink.kafka.security.protocol**: The value is **SASL_PLAINTEXT** for a security cluster and **PLAINTEXT** for a normal cluster.
- **client.sinks.kafka_sink.kafka.kerberos.domain.name**:

  You do not need to set this parameter for a normal cluster. For a security cluster, the value of this parameter is the value of **kerberos.domain.name** in the Kafka cluster.

  In the preceding paths, **X** indicates a random number. Change it based on site requirements. The file must be saved by the user who installs the Flume client, for example, user **root**.

Step 10 After the parameters are set and saved, the Flume client automatically loads the content configured in **properties.properties**. When new log files are generated by spoolDir, the files are sent to Kafka producers and can be consumed by Kafka consumers.

----End

7.5 Configuring a Non-Encrypted Flume Data Collection Task

7.5.1 Generating Configuration Files for the Flume Server and Client

This section describes how to configure Flume server and client parameters after the cluster and the Flume service are installed to ensure proper running of the service.

NOTE

By default, the cluster network environment is secure and the SSL authentication is not enabled during the data transmission process. For details about how to use the encryption mode, see **Configuring an Encrypted Flume Data Collection Task**.

Generating Flume Configuration Files

Step 1 Install the Flume client.

Step 2 Configure the client parameters of the Flume role.

1. Use the Flume configuration tool on FusionInsight Manager to configure the Flume role client parameters and generate a configuration file.

   a. Log in to FusionInsight Manager. Choose **Cluster** > **Services** > **Flume** > **Configuration Tool**.

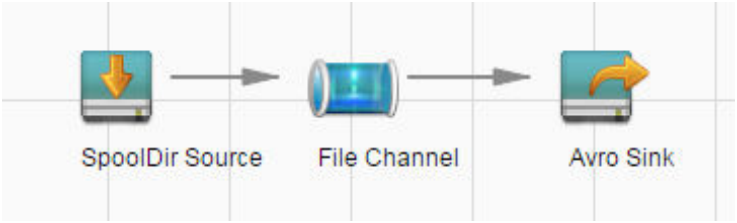
   Figure 7-2 Clicking Configuration Tool

   b. Set **Agent Name** to **client**. Select and drag the source, channel, and sink to be used to the GUI on the right, and connect them.

   For example, use SpoolDir Source, File Channel, and Avro Sink, as shown in **Figure 7-3**.

   Figure 7-3 Example for the Flume configuration tool

   c. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-22** based on the actual environment.

   NOTE

   - If the client parameters of the Flume role have been configured, you can obtain the existing client parameter configuration file from *client installation directory*/**fusioninsight-flume-1.9.0/conf/properties.properties** to ensure that the configuration is in concordance with the previous. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Flume** > **Configuration** > **Import**, import the file, and modify the configuration items related to non-encrypted transmission.
   - It is recommended that the numbers of Sources, Channels, and Sinks do not exceed 40 during configuration file import. Otherwise, the response time may be very long.

Table 7-22 Parameters to be modified for the Flume role client

Parameter	Description	Example Value
ssl	Specifies whether to enable the SSL authentication. (You are advised to enable this function to ensure security.) Only Sources of the Avro type have this configuration item. ■ **true** indicates that the function is enabled. ■ **false** indicates that the function is not enabled.	false

d. Click **Export** to save the **properties.properties** configuration file to the local server.

2. Upload the **properties.properties** file to **flume/conf/** under the installation directory of the Flume client.

Step 3 Configure the server parameters of the Flume role and upload the configuration file to the cluster.

1. Use the Flume configuration tool on the FusionInsight Manager portal to configure the server parameters and generate the configuration file.

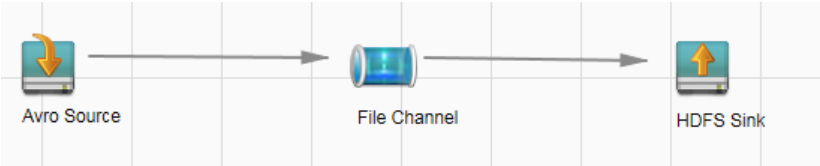
a. Log in to FusionInsight Manager. Choose **Cluster** > **Services** > **Flume** > **Configuration Tool**.

Figure 7-4 Clicking Configuration Tool

b. Set **Agent Name** to **server**. Select and drag the source, channel, and sink to be used to the GUI on the right, and connect them.

For example, use Avro Source, File Channel, and HDFS Sink, as shown in **Figure 7-5**.

Figure 7-5 Example for the Flume configuration tool



c. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-23** based on the actual environment.

NOTE

- If the server parameters of the Flume role have been configured, you can choose **Cluster** > **Services** > **Flume** > **Instance** on FusionInsight Manager. Then select the corresponding Flume role instance and click the **Download** button behind the **flume.config.file** parameter on the **Instance Configurations** page to obtain the existing server parameter configuration file. Choose **Cluster** > **Service** > **Flume** > **Configurations** > **Import**, import the file, and modify the configuration items related to non-encrypted transmission.
- It is recommended that the numbers of Sources, Channels, and Sinks do not exceed 40 during configuration file import. Otherwise, the response time may be very long.
- A unique checkpoint directory needs to be configured for each File Channel.

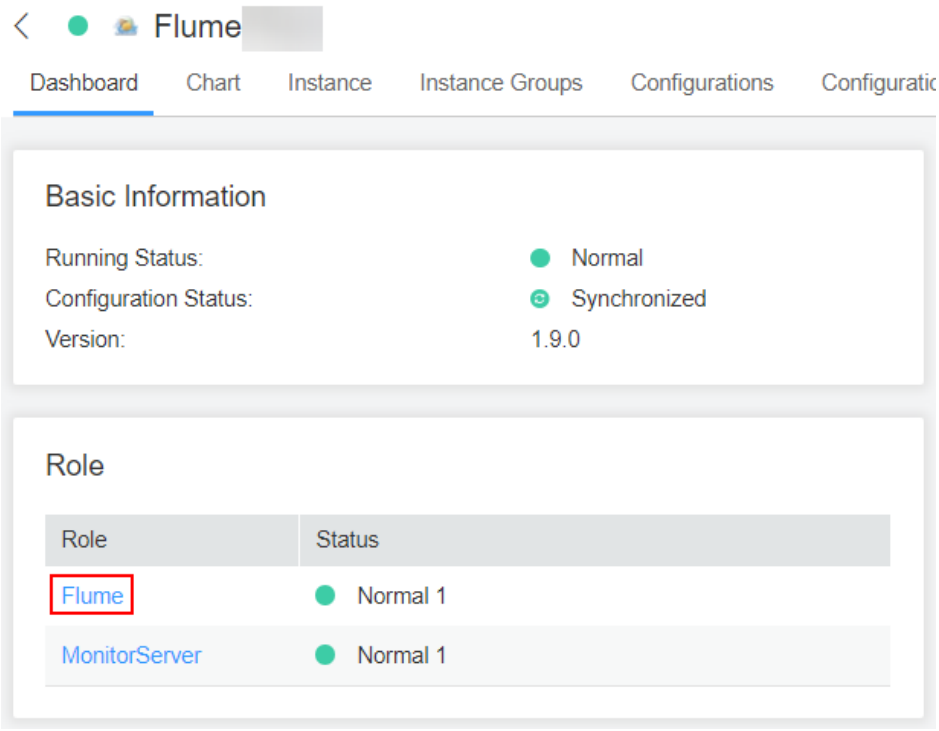
Table 7-23 Parameters to be modified for the Flume role server

Parameter	Description	Example Value
ssl	Specifies whether to enable the SSL authentication. (You are advised to enable this function to ensure security.) Only Sources of the Avro type have this configuration item. ■ **true** indicates that the function is enabled. ■ **false** indicates that the function is not enabled.	false

d. Click **Export** to save the **properties.properties** configuration file to the local server.

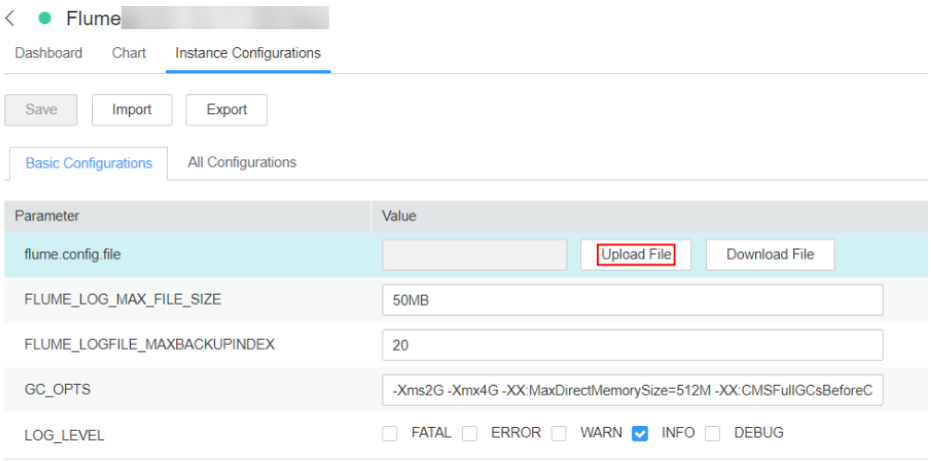
2. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Flume**. In the **Instances** tab, click the Flume role whose configuration file is to be uploaded.

Figure 7-6 Clicking the Flume role



3. On the **Instance Configurations** page, click **Upload File** next to **flume.config.file** and select the **properties.properties** file.

Figure 7-7 Uploading a file



NOTE

- An independent server configuration file can be uploaded to each Flume instance.
- This step is required for updating the configuration file. Modifying the configuration file on the background is an improper operation because the modification will be overwritten after configuration synchronization.

4. Click **Save**, and then click **OK**.
5. Click **Finish**.

----End

7.5.2 Using Flume Server to Collect Static Logs from Local Host to Kafka

This section describes how to use the Flume server to collect static logs from a local host and save them to the topic list (test1) of Kafka.

NOTE

By default, the cluster network environment is secure and the SSL authentication is not enabled during the data transmission process. For details about how to use the encryption mode, see **Configuring an Encrypted Flume Data Collection Task**. The configuration applies to scenarios where only the Flume is configured, for example, Spooldir Source +Memory Channel+Kafka Sink.

Prerequisites

- The cluster has been installed, including the Kafka and Flume services.
- The network environment of the cluster is secure.
- The MRS cluster administrator has understood service requirements and prepared Kafka administrator **flume_kafka**.

Procedure

Step 1 Set Flume parameters.

Use the Flume configuration tool on Manager to configure the Flume role server parameters and generate a configuration file.

1. Log in to FusionInsight Manager. Choose **Cluster** > **Services** > **Flume** > **Configuration Tool**.

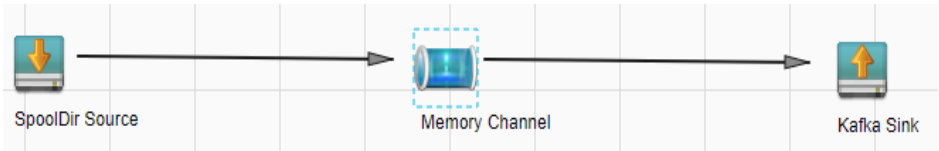
   Figure 7-8 Choosing **Configuration Tool**

2. Set **Agent Name** to **server**. Select and drag the source, channel, and sink to be used to the GUI on the right, and connect them.

   Use SpoolDir Source, Memory Channel, and Kafka Sink.

   Figure 7-9 Example for the Flume configuration tool

3. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-24** based on the actual environment.

NOTE

- If you want to continue using the **properties.propretites** file by modifying it, log in to FusionInsight Manager, choose **Cluster** > **Services**. On the page that is displayed, choose **Flume**. On the displayed page, click the **Configuration Tool** tab, click **Import**, import the file, and modify the configuration items related to non-encrypted transmission.
- It is recommended that the numbers of Sources, Channels, and Sinks do not exceed 40 during configuration file import. Otherwise, the response time may be very long.

Table 7-24 Parameters to be modified for the Flume role server

Parameter	Description	Example Value
Name	The value must be unique and cannot be left blank.	test
spoolDir	Specifies the directory where the file to be collected resides. This parameter cannot be left blank. The directory needs to exist and have the write, read, and execute permissions on the flume running user.	/srv/BigData/hadoop/data1/zb
trackerDir	Specifies the path for storing the metadata of files collected by Flume.	/srv/BigData/hadoop/data1/tracker
batchSize	Specifies the number of events that Flume sends in a batch (number of data pieces). A larger value indicates higher performance and lower timeliness.	61200
kafka.topics	Specifies the list of subscribed Kafka topics, which are separated by commas (,). This parameter cannot be left blank.	test1
kafka.bootstrap.servers	Specifies the bootstrap IP address and port list of Kafka. The default value is all Kafkabrokers in the Kafka cluster.	192.168.101.10:21007

4. Click **Export** to save the **properties.properties** configuration file to the local server.

Step 2 Upload the configuration file.

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Flume**. In the **Instances** tab, click the Flume role whose configuration file is to be uploaded. On the **Instance Configurations** page, click **Upload File** on the right of **flume.config.file**, select the **properties.properties** file exported in **Step 1.4**.

Step 3 Verify log transmission.

1. Log in to the Kafka client.

   cd *Kafka client installation directory*/**Kafka/kafka**

   kinit flume_kafka (Enter the password.)

2. Read data from a Kafka topic.

   bin/kafka-console-consumer.sh --topic *topic name* **--bootstrap-server** *Kafka service IP address of the node where the role instance is located*: **21007 --consumer.config config/consumer.properties --from-beginning**

   The system displays the contents of the file to be collected.

   ```
   [root@host1 kafka]# bin/kafka-console-consumer.sh --topic test1 --bootstrap-server 192.168.101.10:21007 --consumer.config config/consumer.properties --from-beginning
   Welcome to flume
   ```

----End

7.5.3 Using Flume Server to Collect Static Logs from Local Host to HDFS

This section describes how to use the Flume server to collect static logs from a local host and save them to the **/flume/test** directory on HDFS.

NOTE

By default, the cluster network environment is secure and the SSL authentication is not enabled during the data transmission process. For details about how to use the encryption mode, see **Configuring an Encrypted Flume Data Collection Task**. The configuration applies to scenarios where only the Flume is configured, for example, Spooldir Source +Memory Channel+HDFS Sink.

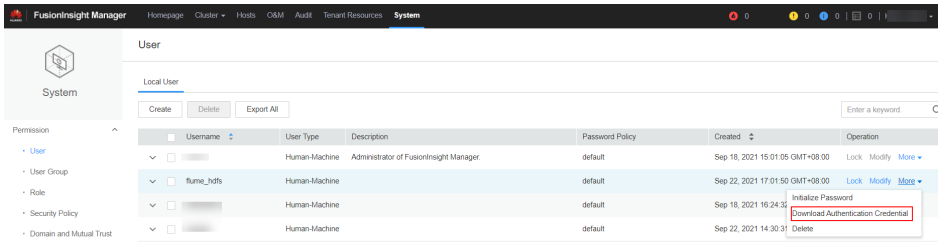
Prerequisites

- The cluster has been installed, including the HDFS and Flume services.
- The network environment of the cluster is secure.
- User **flume_hdfs** has been created, and the HDFS directory and data used for log verification have been authorized to the user.

Procedure

Step 1 On FusionInsight Manager, choose **System** > **Permission** > **User**, select user **flume_hdfs**, and choose **More** > **Download Authentication Credential** to download the Kerberos certificate file of user **flume_hdfs** and save it to the local host.

Figure 7-10 Downloading the authentication credential



Step 2 Set Flume parameters.

Use Flume on FusionInsight Manager to configure the Flume role server parameters and generate a configuration file.

1. Log in to FusionInsight Manager. Choose **Cluster** > **Services** > **Flume** > **Configuration Tool**.

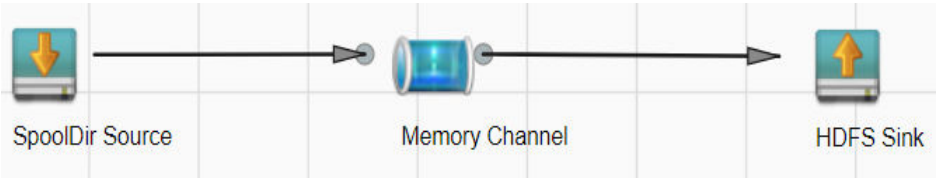
   Figure 7-11 Choosing **Configuration Tool**

2. Set **Agent Name** to **server**. Select the source, channel, and sink to be used, drag them to the GUI on the right, and connect them.

   Use SpoolDir Source, Memory Channel, and HDFS Sink.

   Figure 7-12 Example for the Flume configuration tool

3. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-25** based on the actual environment.

   NOTE

   - If you want to continue using the **properties.propretites** file by modifying it, log in to FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services**. On the page that is displayed, choose **Flume**. On the displayed page, click the **Configuration Tool** tab, click **Import**, import the file, and modify the configuration items related to non-encrypted transmission.
   - It is recommended that the numbers of Sources, Channels, and Sinks do not exceed 40 during configuration file import. Otherwise, the response time may be very long.

Table 7-25 Parameters to be modified for the Flume role server

Parameter	Description	Example Value
Name	The value must be unique and cannot be left blank.	test
spoolDir	Specifies the directory where the file to be collected resides. This parameter cannot be left blank. The directory needs to exist and have the write, read, and execute permissions on the flume running user.	/srv/BigData/hadoop/data1/zb
trackerDir	Specifies the path for storing the metadata of files collected by Flume.	/srv/BigData/hadoop/data1/tracker
batchSize	Specifies the number of events that Flume sends in a batch.	61200
hdfs.path	Specifies the HDFS data write directory. This parameter cannot be left blank.	hdfs://hacluster/flume/test
hdfs.filePrefix	Specifies the file name prefix after data is written to HDFS.	TMP_
hdfs.batchSize	Specifies the maximum number of events that can be written to HDFS once.	61200
hdfs.kerberosPrincipal	Specifies the Kerberos authentication user, which is mandatory in security versions. This configuration is required only in security clusters.	flume_hdfs
hdfs.kerberosKeytab	Specifies the keytab file path for Kerberos authentication, which is mandatory in security versions. This configuration is required only in security clusters.	/opt/test/conf/user.keytab **NOTE** Obtain the **user.keytab** file from the Kerberos certificate file of the user **flume_hdfs**. In addition, ensure that the user who installs and runs the Flume client has the read and write permissions on the **user.keytab** file.

Parameter	Description	Example Value
hdfs.useLocalTimeStamp	Specifies whether to use the local time. Possible values are **true** and **false**.	true

4. Click **Export** to save the **properties.properties** configuration file to the local.

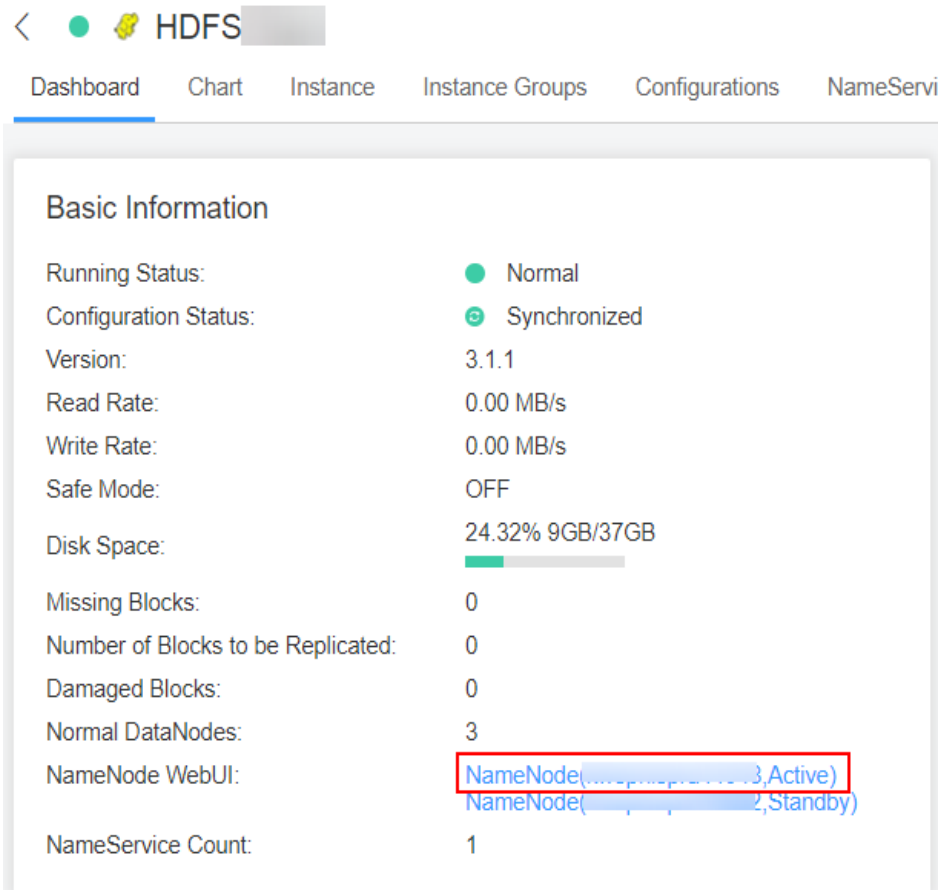
Step 3 Upload the configuration file.

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Flume**. In the **Instances** tab, click the Flume role whose configuration file is to be uploaded. On the **Instance Configurations** page, click **Upload File** on the right of **flume.config.file**, select the **properties.properties** file exported in **Step 2.4**.

Step 4 Verify log transmission.

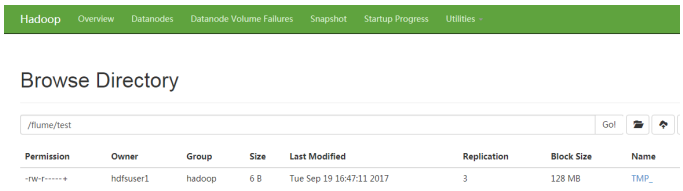
1. Log in to FusionInsight Manager as a user who has the management permission on HDFS. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **HDFS**. On the page that is displayed, click the **NameNode(*Node name*,Active)** link next to **NameNode WebUI** to go to the HDFS web UI. On the displayed page, choose **Utilities** > **Browse the file system**.

Figure 7-13 Accessing the HDFS WebUI



2. Check whether the data is generated in the **/flume/test** directory on the HDFS.

Figure 7-14 Checking HDFS directories and files



----End

7.5.4 Using Flume Server to Collect Dynamic Logs from Local Host to HDFS

This section describes how to use the Flume server to collect dynamic logs from a local host and save them to the **/flume/test** directory on HDFS.

NOTE

By default, the cluster network environment is secure and the SSL authentication is not enabled during the data transmission process. For details about how to use the encryption mode, see **Configuring an Encrypted Flume Data Collection Task**. The configuration applies to scenarios where only the Flume is configured, for example, Taildir Source +Memory Channel+HDFS Sink.

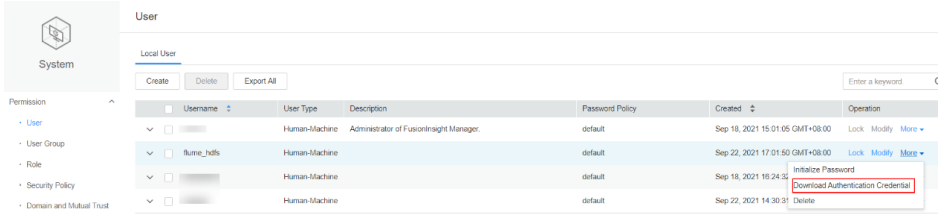
Prerequisites

- The cluster has been installed, including the HDFS and Flume services.
- The network environment of the cluster is secure.
- You have created user **flume_hdfs** and authorized the HDFS directory and data to be operated during log verification.

Procedure

Step 1 On FusionInsight Manager, choose **System > User** and choose **More > Download Authentication Credential** to download the Kerberos certificate file of user **flume_hdfs** and save it to the local host.

Figure 7-15 Downloading the authentication credential



Step 2 Set Flume parameters.

Use the Flume configuration tool on FusionInsight Manager to configure the Flume role server parameters and generate a configuration file.

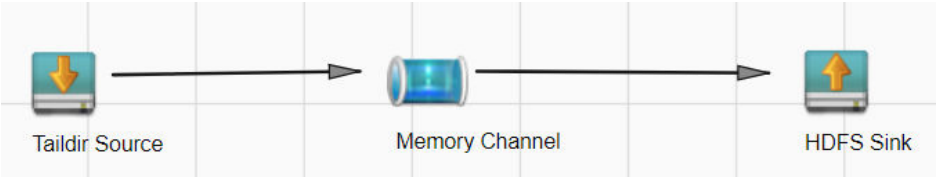
1. Log in to FusionInsight Manager and choose **Cluster** > **Services**. On the page that is displayed, choose **Flume**. On the displayed page, click the **Configuration Tool** tab.

Figure 7-16 Choosing **Configuration Tool**

2. Set **Agent Name** to **server**. Select the source, channel, and sink to be used, drag them to the GUI on the right, and connect them.

   Use Taildir Source, Memory Channel, and HDFS Sink.

   Figure 7-17 Example for the Flume configuration tool

3. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-26** based on the actual environment.

   NOTE

   - If you want to continue using the **properties.propretites** file by modifying it, log in to FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services**. On the page that is displayed, choose **Flume**. On the displayed page, click the **Configuration Tool** tab, click **Import**, import the file, and modify the configuration items related to non-encrypted transmission.
   - It is recommended that the numbers of Sources, Channels, and Sinks do not exceed 40 during configuration file import. Otherwise, the response time may be very long.

Table 7-26 Parameters to be modified for the Flume role server

Parameter	Description	Example Value
Name	The value must be unique and cannot be left blank.	test
filegroups	Specifies the file group list name. This parameter cannot be left blank. The value contains the following two parts: – **Name**: name of the file group list. – **filegroups**: absolute path of dynamic log files.	-

Parameter	Description	Example Value
positionFile	Specifies the location where the collected file information (file name and location from which the file collected) is saved. This parameter cannot be left blank. The file does not need to be created manually, but the Flume running user needs to have the write permission on its upper-level directory.	/home/omm/flume/positionfile
batchSize	Specifies the number of events that Flume sends in a batch.	61200
hdfs.path	Specifies the HDFS data write directory. This parameter cannot be left blank.	hdfs://hacluster/flume/test
hdfs.filePrefix	Specifies the file name prefix after data is written to HDFS.	TMP_
hdfs.batchSize	Specifies the maximum number of events that can be written to HDFS once.	61200
hdfs.kerberosPrincipal	Specifies the Kerberos authentication user, which is mandatory in security versions. This configuration is required only in security clusters.	flume_hdfs
hdfs.kerberosKeytab	Specifies the keytab file path for Kerberos authentication, which is mandatory in security versions. This configuration is required only in security clusters.	/opt/test/conf/user.keytab **NOTE** Obtain the **user.keytab** file from the Kerberos certificate file of the user **flume_hdfs**. In addition, ensure that the user who installs and runs the Flume client has the read and write permissions on the **user.keytab** file.
hdfs.useLocalTimeStamp	Specifies whether to use the local time. Possible values are **true** and **false**.	true

4. Click **Export** to save the **properties.properties** configuration file to the local.

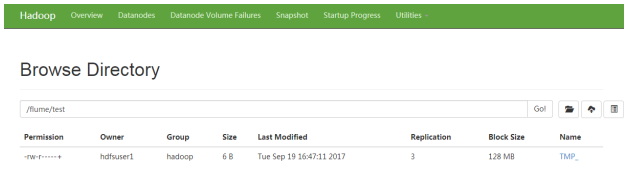
Step 3 Upload the configuration file.

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Flume**. In the **Instances** tab, click the Flume role whose configuration file is to be uploaded. On the **Instance Configurations** page, click **Upload File** on the right of **flume.config.file**, select the **properties.properties** file exported in **Step 2.4**.

Step 4 Verify log transmission.

1. Log in to FusionInsight Manager as a user who has the management permission on HDFS. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **HDFS**. On the page that is displayed, click the **NameNode(***Node name***,Active)** link next to **NameNode WebUI** to go to the HDFS web UI. On the displayed page, choose **Utilities** > **Browse the file system**.
2. Check whether the data is generated in the **/flume/test** directory on the HDFS.

Figure 7-18 Checking HDFS directories and files



----End

7.5.5 Using Flume Server to Collect Logs from Kafka to HDFS

This section describes how to use the Flume server to collect logs from the topic list (test1) of Kafka and save them to the **/flume/test** directory on HDFS.

NOTE

By default, the cluster network environment is secure and the SSL authentication is not enabled during the data transmission process. For details about how to use the encryption mode, see **Configuring an Encrypted Flume Data Collection Task**. The configuration applies to scenarios where only the Flume is configured, for example, Kafka Source +Memory Channel+HDFS Sink.

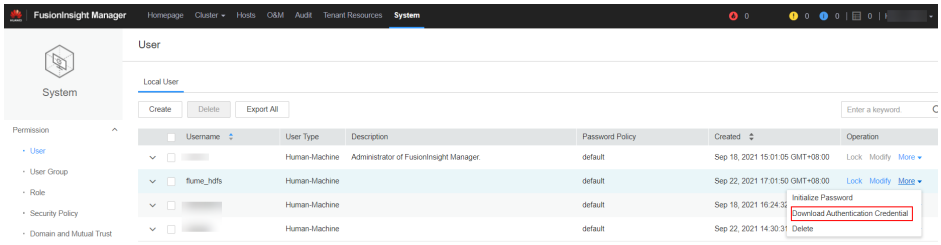
Prerequisites

- The cluster has been installed, including the HDFS, Kafka, and Flume services.
- The network environment of the cluster is secure.
- You have created user **flume_hdfs** and authorized the HDFS directory and data to be operated during log verification.

Procedure

Step 1 On FusionInsight Manager, choose **System > User** and choose **More > Download Authentication Credential** to download the Kerberos certificate file of user **flume_hdfs** and save it to the local host.

Figure 7-19 Downloading the authentication credential

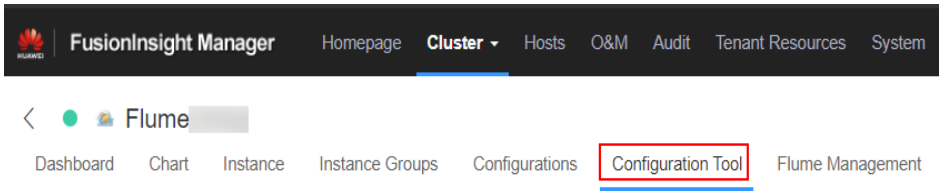


Step 2 Configure the server parameters of the Flume role.

Use the Flume configuration tool on FusionInsight Manager to configure the Flume role server parameters and generate a configuration file.

1. Log in to FusionInsight Manager and choose **Cluster** > **Services**. On the page that is displayed, choose **Flume**. On the displayed page, click the **Configuration Tool** tab.

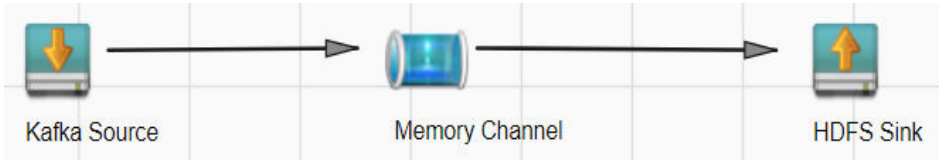
   Figure 7-20 Choosing **Configuration Tool**

2. Set **Agent Name** to **server**. Select the source, channel, and sink to be used, drag them to the GUI on the right, and connect them.

   For example, use Kafka Source, Memory Channel, and HDFS Sink.

   Figure 7-21 Example for the Flume configuration tool

3. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-27** based on the actual environment.

   NOTE

   - If you want to continue using the **properties.propretites** file by modifying it, log in to FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services**. On the page that is displayed, choose **Flume**. On the displayed page, click the **Configuration Tool** tab, click **Import**, import the file, and modify the configuration items related to non-encrypted transmission.
   - It is recommended that the numbers of Sources, Channels, and Sinks do not exceed 40 during configuration file import. Otherwise, the response time may be very long.

Table 7-27 Parameters to be modified for the Flume role server

Parameter	Description	Example Value
Name	The value must be unique and cannot be left blank.	test
kafka.topics	Specifies the subscribed Kafka topic list, in which topics are separated by commas (,). This parameter cannot be left blank.	test1
kafka.consumer.group.id	Specifies the data group ID obtained from Kafka. This parameter cannot be left blank.	flume
kafka.bootstrap.servers	Specifies the bootstrap IP address and port list of Kafka. The default value is all Kafka lists in a Kafka cluster. If Kafka has been installed in the cluster and its configurations have been synchronized, this parameter can be left blank.	192.168.101.10:9092
batchSize	Specifies the number of events that Flume sends in a batch (number of data pieces).	61200
hdfs.path	Specifies the HDFS data write directory. This parameter cannot be left blank.	hdfs://hacluster/flume/test
hdfs.filePrefix	Specifies the file name prefix after data is written to HDFS.	TMP_
hdfs.batchSize	Specifies the maximum number of events that can be written to HDFS once.	61200
hdfs.kerberosPrincipal	Specifies the Kerberos authentication user, which is mandatory in security versions. This configuration is required only in security clusters.	flume_hdfs

Parameter	Description	Example Value
hdfs.kerberosKeytab	Specifies the keytab file path for Kerberos authentication, which is mandatory in security versions. This configuration is required only in security clusters.	/opt/test/conf/user.keytab **NOTE** Obtain the **user.keytab** file from the Kerberos certificate file of the user **flume_hdfs**. In addition, ensure that the user who installs and runs the Flume client has the read and write permissions on the **user.keytab** file.
hdfs.useLocalTimeStamp	Specifies whether to use the local time. Possible values are **true** and **false**.	true

4. Click **Export** to save the **properties.properties** configuration file to the local.

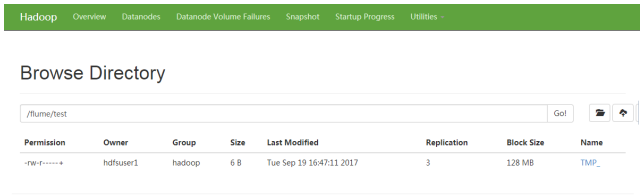
Step 3 Upload the configuration file.

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Flume**. In the **Instances** tab, click the Flume role whose configuration file is to be uploaded. On the **Instance Configurations** page, click **Upload File** on the right of **flume.config.file**, select the **properties.properties** file exported in **Step 2.4**.

Step 4 Verify log transmission.

1. Log in to FusionInsight Manager as a user who has the management permission on HDFS. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **HDFS**. On the page that is displayed, click the **NameNode(***Node name***,Active)** link next to **NameNode WebUI** to go to the HDFS web UI. On the displayed page, choose **Utilities** > **Browse the file system**.
2. Check whether the data is generated in the **/flume/test** directory on the HDFS.

Figure 7-22 Checking HDFS directories and files



----End

7.5.6 Using Flume Client to Collect Logs from Kafka to HDFS

This section describes how to use the Flume client to collect logs from the topic list (test1) of the Kafka client and save them to the **/flume/test** directory on HDFS.

NOTE

By default, the cluster network environment is secure and the SSL authentication is not enabled during the data transmission process. For details about how to use the encryption mode, see **Configuring an Encrypted Flume Data Collection Task**.

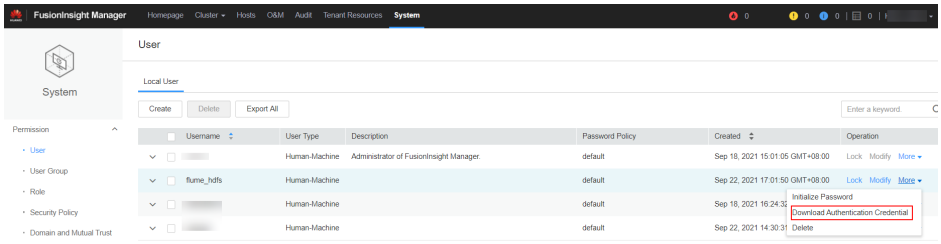
Prerequisites

- The Flume client has been installed.
- The cluster has been installed, including the HDFS, Kafka, and Flume services.
- You have created user **flume_hdfs** and authorized the HDFS directory and data to be operated during log verification.
- The network environment of the cluster is secure.

Procedure

Step 1 On FusionInsight Manager, choose **System > User** and choose **More > Download Authentication Credential** to download the Kerberos certificate file of user **flume_hdfs** and save it to the local host.

Figure 7-23 Downloading the authentication credential

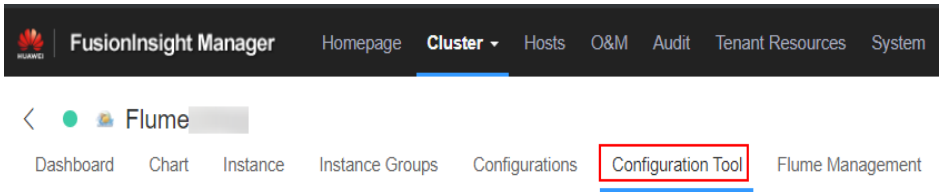


Step 2 Configure the client parameters of the Flume role.

1. Use the Flume configuration tool on FusionInsight Manager to configure the Flume role server parameters and generate a configuration file.

   a. Log in to FusionInsight Manager and choose **Cluster** > **Services**. On the page that is displayed, choose **Flume**. On the displayed page, click the **Configuration Tool** tab.

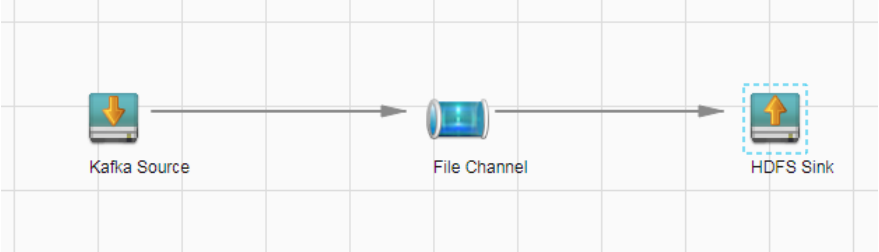
   Figure 7-24 Choosing **Configuration Tool**

   b. Set **Agent Name** to **client**. Select the source, channel, and sink to be used, drag them to the GUI on the right, and connect them.

   For example, use Kafka Source, File Channel, and HDFS Sink, as shown in **Figure 7-25**.

Figure 7-25 Example for the Flume configuration tool



c. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-28** based on the actual environment.

NOTE

- If you want to continue using the **properties.propretites** file by modifying it, log in to FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services**. On the page that is displayed, choose **Flume**. On the displayed page, click the **Configuration Tool** tab, click **Import**, import the file, and modify the configuration items related to non-encrypted transmission.
- It is recommended that the numbers of Sources, Channels, and Sinks do not exceed 40 during configuration file import. Otherwise, the response time may be very long.

Table 7-28 Parameters to be modified for the Flume role client

Parameter	Description	Example Value
Name	The value must be unique and cannot be left blank.	test
kafka.topics	Specifies the subscribed Kafka topic list, in which topics are separated by commas (,). This parameter cannot be left blank.	test1
kafka.consumer.group.id	Specifies the data group ID obtained from Kafka. This parameter cannot be left blank.	flume

Parameter	Description	Example Value
kafka.bootstrap.servers	Specifies the bootstrap IP address and port list of Kafka. The default value is all Kafka lists in a Kafka cluster. If Kafka has been installed in the cluster and its configurations have been synchronized, this parameter can be left blank.	192.168.101.10:21007
batchSize	Specifies the number of events that Flume sends in a batch (number of data pieces).	61200
dataDirs	Specifies the directory for storing buffer data. The run directory is used by default. Configuring multiple directories on disks can improve transmission efficiency. Use commas (,) to separate multiple directories. If the directory is inside the cluster, the **/srv/BigData/hadoop/dataX/flume/data** directory can be used. **dataX** ranges from data1 to dataN. If the directory is outside the cluster, it needs to be independently planned.	/srv/BigData/hadoop/data1/flume/data

Parameter	Description	Example Value
checkpointDir	Specifies the directory for storing the checkpoint information, which is under the run directory by default. If the directory is inside the cluster, the **/srv/BigData/hadoop/dataX/flume/checkpoint** directory can be used. **dataX** ranges from data1 to dataN. If the directory is outside the cluster, it needs to be independently planned.	/srv/BigData/hadoop/data1/flume/checkpoint
transactionCapacity	Specifies the transaction size, that is, the number of events in a transaction that can be processed by the current Channel. The size cannot be smaller than the batchSize of Source. Setting the same size as batchSize is recommended.	61200
hdfs.path	Specifies the HDFS data write directory. This parameter cannot be left blank.	hdfs://hacluster/flume/test
hdfs.filePrefix	Specifies the file name prefix after data is written to HDFS.	TMP_
hdfs.batchSize	Specifies the maximum number of events that can be written to HDFS once.	61200

Parameter	Description	Example Value
hdfs.kerberosPrincipal	Specifies the Kerberos authentication user, which is mandatory in security versions. This configuration is required only in security clusters.	flume_hdfs
hdfs.kerberosKeytab	Specifies the keytab file path for Kerberos authentication, which is mandatory in security versions. This configuration is required only in security clusters.	/opt/test/conf/user.keytab **NOTE** Obtain the **user.keytab** file from the Kerberos certificate file of the user **flume_hdfs**. In addition, ensure that the user who installs and runs the Flume client has the read and write permissions on the **user.keytab** file.
hdfs.useLocalTimeStamp	Specifies whether to use the local time. Possible values are **true** and **false**.	true

d. Click **Export** to save the **properties.properties** configuration file to the local.

2. Upload the **properties.properties** file to **flume/conf/** in the Flume client installation directory.
3. To connect the Flume client to the HDFS, you need to add the following configuration:

   a. Download the Kerberos certificate of account **flume_hdfs** and obtain the **krb5.conf** configuration file. Upload the configuration file to the **fusioninsight-flume-1.9.0/conf/** directory on the node where the client is installed.

   b. In **fusioninsight-flume-1.9.0/conf/**, create the **jaas.conf** configuration file.

      vi jaas.conf

      ```
      KafkaClient {
      com.sun.security.auth.module.Krb5LoginModule required
      useKeyTab=true
      keyTab="/opt/test/conf/user.keytab"
      principal="flume_hdfs@<System domain name>"
      useTicketCache=false
      storeKey=true
      debug=true;
      };
      ```

      Values of **keyTab** and **principal** vary depending on the actual situation.

c. Obtain configuration files **core-site.xml** and **hdfs-site.xml** from **/opt/FusionInsight_Cluster_*<Cluster ID>*_Flume_ClientConfig/Flume/config** and upload them to **fusioninsight-flume-1.9.0/conf/**.

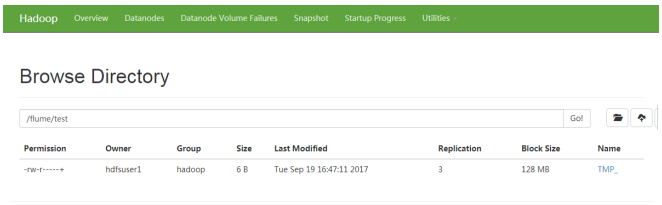
4. Go to **fusioninsight-flume-1.9.0/bin** in the installation directory of the client node and run the following command to restart the Flume process:

   ./flume-manage.sh restart

Step 3 Verify log transmission.

1. Log in to FusionInsight Manager as a user who has the management permission on HDFS. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **HDFS**. On the page that is displayed, click the **NameNode(*Node name*,Active)** link next to **NameNode WebUI** to go to the HDFS web UI. On the displayed page, choose **Utilities** > **Browse the file system**.
2. Check whether the data is generated in the **/flume/test** directory on the HDFS.

Figure 7-26 Checking HDFS directories and files



----End

7.5.7 Using Cascaded Agents to Collect Static Logs from Local Host to HBase

This section describes how to use the Flume client to collect static logs from a local host and save them to the **flume_test** HBase table. In this scenario, multi-level agents are cascaded.

NOTE

By default, the cluster network environment is secure and the SSL authentication is not enabled during the data transmission process. For details about how to use the encryption mode, see **Configuring an Encrypted Flume Data Collection Task**. The configuration applies to scenarios where only the server is configured, for example, Spooldir Source+File Channel+HBase Sink.

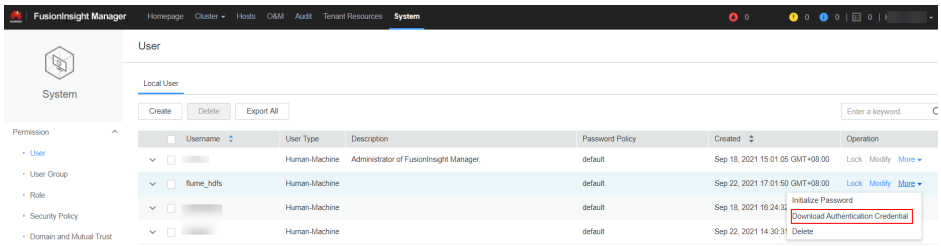
Prerequisites

- The cluster has been installed, including the HBase and Flume services.
- The Flume client has been installed.
- The network environment of the cluster is secure.
- An HBase table has been created by running the **create 'flume_test', 'cf'** command.
- The MRS cluster administrator has understood service requirements and prepared HBase administrator **flume_hbase**.

Procedure

Step 1 On FusionInsight Manager, choose **System > User** and choose **More > Download Authentication Credential** to download the Kerberos certificate file of user **flume_hbase** and save it to the local host.

Figure 7-27 Downloading the authentication credential

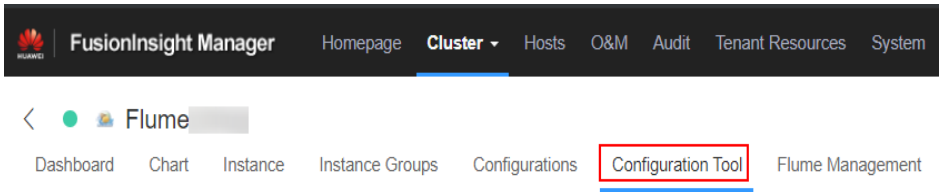


Step 2 Configure the client parameters of the Flume role.

1. Use the Flume configuration tool on FusionInsight Manager to configure the Flume role client parameters and generate a configuration file.

   a. Log in to FusionInsight Manager and choose **Cluster** > **Services**. On the page that is displayed, choose **Flume**. On the displayed page, click the **Configuration Tool** tab.

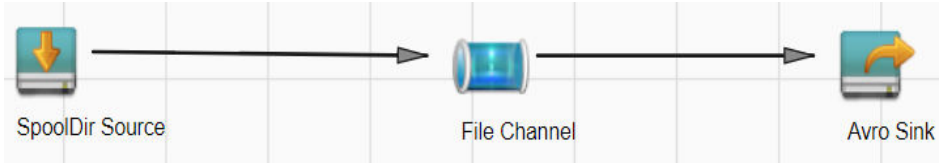
   Figure 7-28 Choosing **Configuration Tool**

   b. Set **Agent Name** to **client**. Select the source, channel, and sink to be used, drag them to the GUI on the right, and connect them.

   Use SpoolDir Source, File Channel, and Avro Sink.

   Figure 7-29 Example for the Flume configuration tool

   c. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-29** based on the actual environment.

NOTE

- If you want to continue using the **properties.propretites** file by modifying it, log in to FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services**. On the page that is displayed, choose **Flume**. On the displayed page, click the **Configuration Tool** tab, click **Import**, import the file, and modify the configuration items related to non-encrypted transmission.
- It is recommended that the numbers of Sources, Channels, and Sinks do not exceed 40 during configuration file import. Otherwise, the response time may be very long.

Table 7-29 Parameters to be modified for the Flume role client

Parameter	Description	Example Value
Name	The value must be unique and cannot be left blank.	test
spoolDir	Specifies the directory where the file to be collected resides. This parameter cannot be left blank. The directory needs to exist and have the write, read, and execute permissions on the flume running user.	/srv/BigData/hadoop/data1/zb
trackerDir	Specifies the path for storing the metadata of files collected by Flume.	/srv/BigData/hadoop/data1/tracker
batchSize	Specifies the number of events that Flume sends in a batch (number of data pieces). A larger value indicates higher performance and lower timeliness.	61200

Parameter	Description	Example Value
dataDirs	Specifies the directory for storing buffer data. The run directory is used by default. Configuring multiple directories on disks can improve transmission efficiency. Use commas (,) to separate multiple directories. If the directory is inside the cluster, the **/srv/BigData/hadoop/dataX/flume/data** directory can be used. **dataX** ranges from data1 to dataN. If the directory is outside the cluster, it needs to be independently planned.	/srv/BigData/hadoop/data1/flume/data
checkpointDir	Specifies the directory for storing the checkpoint information, which is under the run directory by default. If the directory is inside the cluster, the **/srv/BigData/hadoop/dataX/flume/checkpoint** directory can be used. **dataX** ranges from data1 to dataN. If the directory is outside the cluster, it needs to be independently planned.	/srv/BigData/hadoop/data1/flume/checkpoint

Parameter	Description	Example Value
transactionCapacity	Specifies the transaction size, that is, the number of events in a transaction that can be processed by the current Channel. The size cannot be smaller than the batchSize of Source. Setting the same size as batchSize is recommended.	61200
hostname	Specifies the name or IP address of the host whose data is to be sent. This parameter cannot be left blank. Name or IP address must be configured to be the name or IP address that the Avro source associated with it.	192.168.108.11
port	Specifies the port that sends the data. This parameter cannot be left blank. It must be consistent with the port that is monitored by the connected Avro Source.	21154

Parameter	Description	Example Value
ssl	Specifies whether to enable the SSL authentication. (You are advised to enable this function to ensure security.) Only Sources of the Avro type have this configuration item. ■ **true** indicates that the function is enabled. ■ **false** indicates that the client authentication function is not enabled.	false

d. Click **Export** to save the **properties.properties** configuration file to the local.

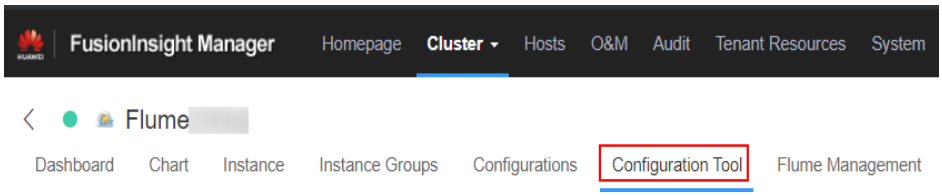
2. Upload the **properties.properties** file to **flume/conf/** under the installation directory of the Flume client.

Step 3 Configure the server parameters of the Flume role and upload the configuration file to the cluster.

1. Use the Flume configuration tool on the FusionInsight Manager portal to configure the server parameters and generate the configuration file.

a. Log in to FusionInsight Manager and choose **Cluster** > **Services**. On the page that is displayed, choose **Flume**. On the displayed page, click the **Configuration Tool** tab.

Figure 7-30 Choosing **Configuration Tool**



b. Set **Agent Name** to **server**. Select the source, channel, and sink to be used, drag them to the GUI on the right, and connect them.

For example, use Avro Source, File Channel, and HBase Sink.

Figure 7-31 Example for the Flume configuration tool

c. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-30** based on the actual environment.

NOTE

- If the server parameters of the Flume role have been configured, you can choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume** > **Instance** on FusionInsight Manager. Then select the corresponding Flume role instance and click the **Download** button behind the **flume.config.file** parameter on the **Instance Configurations** page to obtain the existing server parameter configuration file. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume** > **Configuration Tool** > **Import**, import the file, and modify the configuration items related to non-encrypted transmission.
- It is recommended that the numbers of Sources, Channels, and Sinks do not exceed 40 during configuration file import. Otherwise, the response time may be very long.
- A unique checkpoint directory needs to be configured for each File Channel.

Table 7-30 Parameters to be modified for the Flume role server

Parameter	Description	Example Value
Name	The value must be unique and cannot be left blank.	test
bind	Specifies the IP address to which Avro Source is bound. This parameter cannot be left blank. It must be configured as the IP address that the server configuration file will upload.	192.168.108.11
port	Specifies the ID of the port that the Avro Source monitors. This parameter cannot be left blank. It must be configured as an unused port.	21154

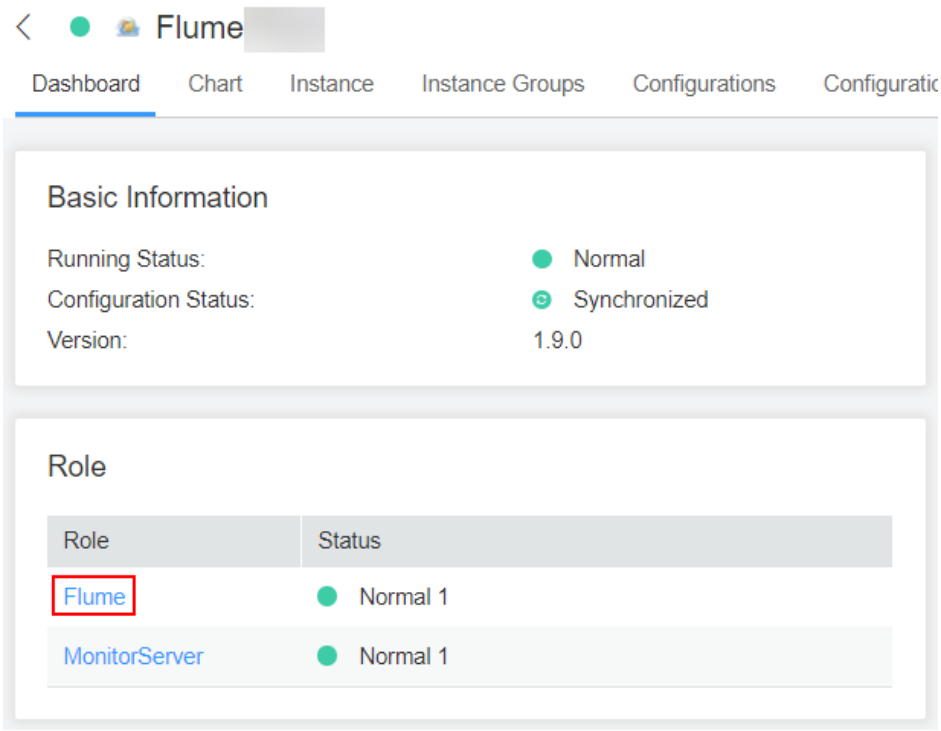
Parameter	Description	Example Value
ssl	Specifies whether to enable the SSL authentication. (You are advised to enable this function to ensure security.) Only Sources of the Avro type have this configuration item. ▪ **true** indicates that the function is enabled. ▪ **false** indicates that the client authentication function is not enabled.	false
dataDirs	Specifies the directory for storing buffer data. The run directory is used by default. Configuring multiple directories on disks can improve transmission efficiency. Use commas (,) to separate multiple directories. If the directory is inside the cluster, the **/srv/BigData/hadoop/dataX/flume/data** directory can be used. **dataX** ranges from data1 to dataN. If the directory is outside the cluster, it needs to be independently planned.	/srv/BigData/hadoop/data1/flumeserver/data
checkpointDir	Specifies the directory for storing the checkpoint information, which is under the run directory by default. If the directory is inside the cluster, the **/srv/BigData/hadoop/dataX/flume/checkpoint** directory can be used. **dataX** ranges from data1 to dataN. If the directory is outside the cluster, it needs to be independently planned.	/srv/BigData/hadoop/data1/flumeserver/checkpoint
transactionCapacity	Specifies the transaction size, that is, the number of events in a transaction that can be processed by the current Channel. The size cannot be smaller than the batchSize of Source. Setting the same size as batchSize is recommended.	61200

Parameter	Description	Example Value
table	Specifies the HBase table name. This parameter cannot be left blank.	flume_test
columnFamily	Specifies the HBase column family name. This parameter cannot be left blank.	cf
batchSize	Specifies the maximum number of events written to HBase by Flume in a batch.	61200
kerberosPrincipal	Specifies the Kerberos authentication user, which is mandatory in security versions. This configuration is required only in security clusters.	flume_hbase
kerberosKeytab	Specifies the file path for Kerberos authentication, which is mandatory in security versions. This configuration is required only in security clusters.	/opt/test/conf/user.keytab **NOTE** Obtain the **user.keytab** file from the Kerberos certificate file of the user **flume_hbase**. In addition, ensure that the user who installs and runs the Flume client has the read and write permissions on the **user.keytab** file.

   d. Click **Export** to save the **properties.properties** configuration file to the local.

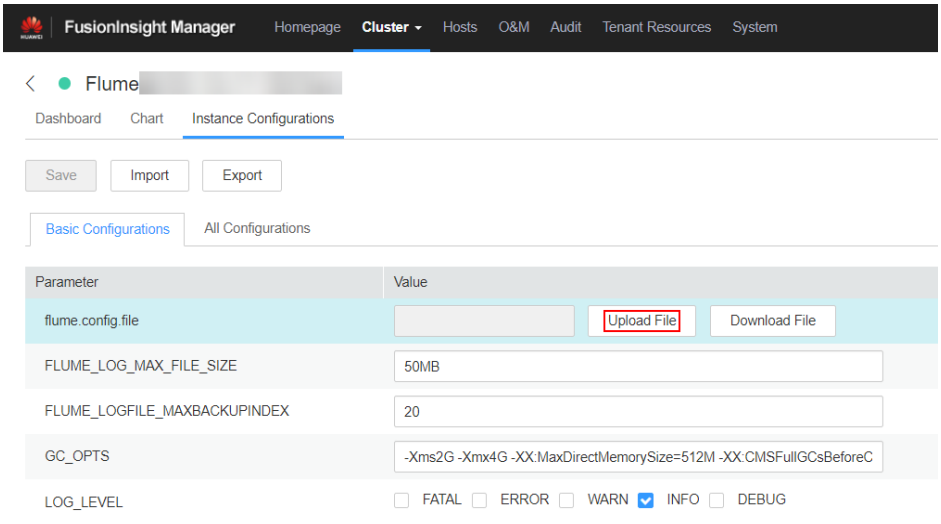
2. Log in to FusionInsight Manager and choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume**. In the **Instances** tab, click the Flume role whose configuration file is to be uploaded.

Figure 7-32 Clicking the Flume role



3. On the **Instance Configurations** page, click **Upload File** next to **flume.config.file** and select the **properties.properties** file.

Figure 7-33 Uploading a file



NOTE

- An independent server configuration file can be uploaded to each Flume instance.
- This step is required for updating the configuration file. Modifying the configuration file on the background is an improper operation because the modification will be overwritten after configuration synchronization.

4. Click **Save**, and then click **OK**.

5. Click **Finish**.

Step 4 Verify log transmission.

1. Go to the directory where the HBase client is installed.

   cd /*Client installation directory***/ HBase/hbase**

   kinit flume_hbase (Enter the password.)
2. Run the **hbase shell** command to access the HBase client.
3. Run the **scan 'flume_test'** statement. Logs are written in the HBase column family by line.

```
hbase(main):001:0> scan 'flume_test'
ROW                                                COLUMN
+CELL

2017-09-18 16:05:36,394 INFO  [hconnection-0x415a3f6a-shared--pool2-t1] ipc.AbstractRpcClient:
RPC Server Kerberos principal name for service=ClientService is hbase/hadoop.<system domain
name>@<system domain name>
 default4021ff4a-9339-4151-a4d0-00f20807e76d                column=cf:pCol,
timestamp=1505721909388, value=Welcome to
flume
 incRow                                                    column=cf:iCol, timestamp=1505721909461, value=
\x00\x00\x00\x00\x00\x00\x00\x01
2 row(s) in 0.3660 seconds
```

----End

7.6 Configuring an Encrypted Flume Data Collection Task

7.6.1 Using Cascaded Agents to Collect Static Logs from Local Host to HDFS

This section describes how to use Flume to collect static logs from a local host and save them to the **/flume/test** directory on HDFS.

Prerequisites

- The cluster, HDFS and Flume services, and Flume client have been installed.
- User **flume_hdfs** has been created, and the HDFS directory and data used for log verification have been authorized to the user.

Procedure

Step 1 Generate the certificate trust lists of the server and client of the Flume role respectively.

1. Log in to the node where the Flume server is located as user **omm**. Go to the **${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/bin** directory.

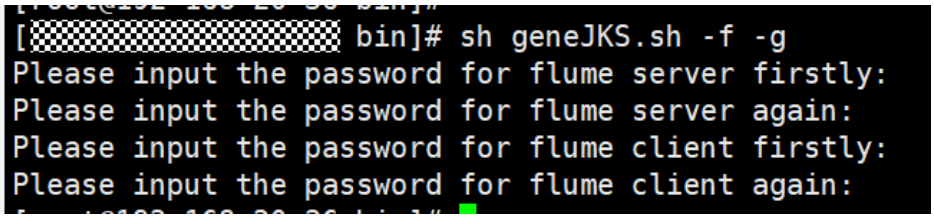
   cd ${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/bin
2. Run the following command to generate and export the server and client certificates of the Flume role:

sh geneJKS.sh -f -g

The generated certificate is saved in the **${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/conf** path .

- **flume_sChat.jks** is the certificate library of the Flume role server. **flume_sChat.crt** is the exported file of the **flume_sChat.jks** certificate. **-f** indicates whether users need to enter the password of the certificate and certificate library.
- **flume_cChat.jks** is the certificate library of the Flume role client. **flume_cChat.crt** is the exported file of the **flume_cChat.jks** certificate. **-g** indicates whether users need to enter the password of the certificate and certificate library.
- **flume_sChatt.jks** and **flume_cChatt.jks** are the SSL certificate trust lists of the Flume server and client, respectively.

Figure 7-34 Example



NOTE

All user-defined passwords involved in this section must meet the following requirements:

- Contain at least four types of the following: uppercase letters, lowercase letters, digits, and special characters.
- Contain at least eight characters and a maximum of 64 characters.
- It is recommended that the user-defined passwords be changed periodically (for example, every three months), and certificates and trust lists be generated again to ensure security.

Step 2 On FusionInsight Manager, choose **System > User** and choose **More > Download Authentication Credential** to download the Kerberos certificate file of user **flume_hdfs** and save it to the local host.

Step 3 Configure the server parameters of the Flume role and upload the configuration file to the cluster.

1. Log in to any node where the Flume role is located as user **omm**. Run the following command to go to the ${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/bin directory:

   cd ${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/bin

2. Run the following command to generate and obtain Flume server keystore password, trust list password, and keystore-password encrypted private key information. Enter the password twice and confirm the password. It is the password of the **flume_sChat.jks** certificate library.

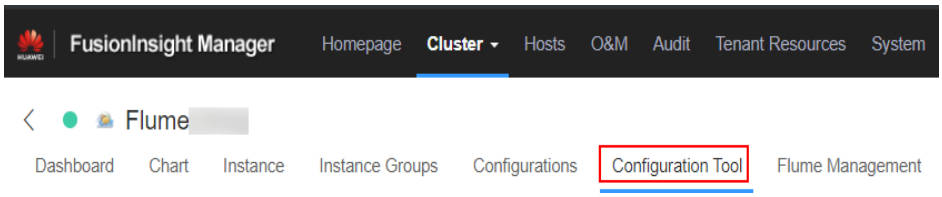
   ./genPwFile.sh

cat password.property

3. Use the Flume configuration tool on the FusionInsight Manager portal to configure the server parameters and generate the configuration file.

   a. Log in to FusionInsight Manager and choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume** > **Configuration Tool**.

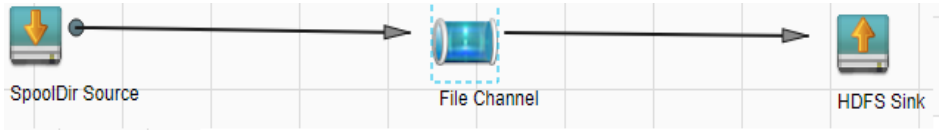
   Figure 7-35 Choosing **Configuration Tool**

   b. Set **Agent Name** to **server**. Select the source, channel, and sink to be used, drag them to the GUI on the right, and connect them.

   For example, use SpoolDir Source, File Channel, and HDFS Sink.

   Figure 7-36 Example for the Flume configuration tool

   c. Double-click the source, channel, and sink. Set corresponding configuration parameters by seeing **Table 7-31** based on the actual environment.

   NOTE

   - If the server parameters of the Flume role have been configured, you can choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume** > **Instance** on FusionInsight Manager. Then select the corresponding Flume role instance and click the **Download** button behind the **flume.config.file** parameter on the **Instance Configurations** page to obtain the existing server parameter configuration file. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume** > **Configuration Tool** > **Import**, import the file, and modify the configuration items related to encrypted transmission.
   - It is recommended that the numbers of Sources, Channels, and Sinks do not exceed 40 during configuration file import. Otherwise, the response time may be very long.
   - A unique checkpoint directory needs to be configured for each File Channel.

   d. Click **Export** to save the **properties.properties** configuration file to the local.

   Table 7-31 Parameters to be modified of the Flume role server

Parameter	Description	Example Value
Name	The value must be unique and cannot be left blank.	test

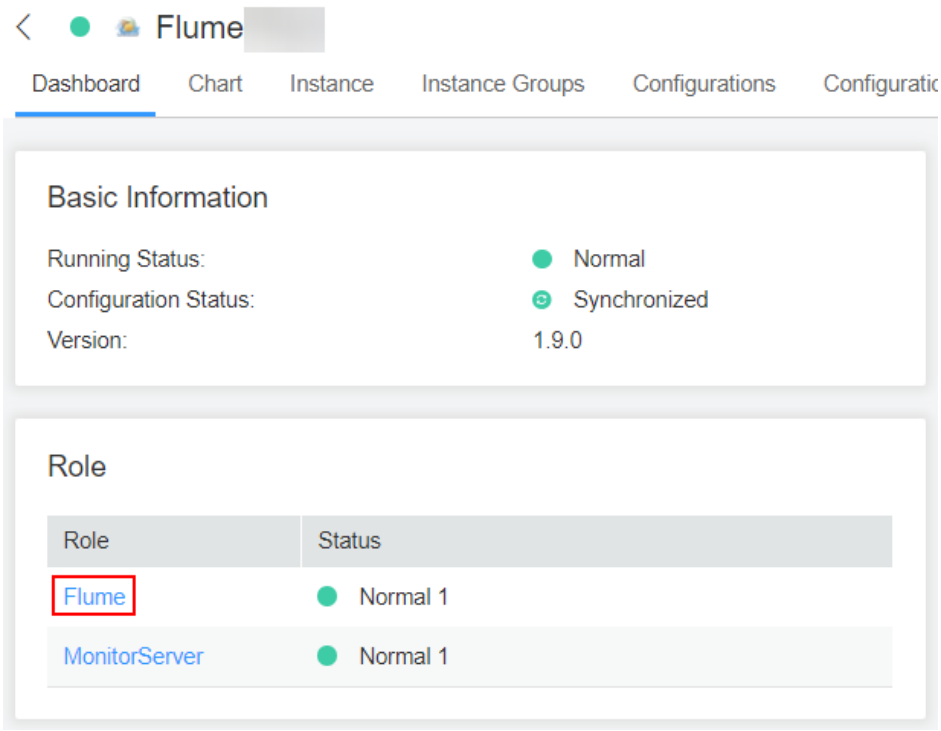
Parameter	Description	Example Value
bind	Specifies the IP address to which Avro Source is bound. This parameter cannot be left blank. It must be configured as the IP address that the server configuration file will upload.	192.168.108.11
port	Specifies the IP address to which Avro Source is bound. This parameter cannot be left blank. It must be configured as an unused port.	21154
ssl	Indicates whether to enable the SSL authentication. (You are advised to enable this function to ensure security.) Only Sources of the Avro type have this configuration item. ■ **true** indicates that the function is enabled. ■ **false** indicates that the client authentication function is not enabled.	true
keystore	Indicates the server certificate.	${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/conf/flume_sChat.jks
keystore-password	Specifies the password of the key library, which is the password required to obtain the keystore information. Enter the value of **password** obtained in **Step 3.2**.	-
truststore	Indicates the SSL certificate trust list of the server.	${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/conf/flume_sChatt.jks

Parameter	Description	Example Value
truststore-password	Specifies the trust list password, which is the password required to obtain the truststore information. Enter the value of **password** obtained in **Step 3.2**.	-
dataDirs	Specifies the directory for storing buffer data. The run directory is used by default. Configuring multiple directories on disks can improve transmission efficiency. Use commas (,) to separate multiple directories. If the directory is inside the cluster, the **/srv/BigData/hadoop/dataX/flume/data** directory can be used. **dataX** ranges from data1 to dataN. If the directory is outside the cluster, it needs to be independently planned.	/srv/BigData/hadoop/data1/flumeserver/data
checkpointDir	Specifies the directory for storing the checkpoint information, which is under the run directory by default. If the directory is inside the cluster, the **/srv/BigData/hadoop/dataX/flume/checkpoint** directory can be used. **dataX** ranges from data1 to dataN. If the directory is outside the cluster, it needs to be independently planned.	/srv/BigData/hadoop/data1/flumeserver/checkpoint
transactionCapacity	Specifies the transaction size, that is, the number of events in a transaction that can be processed by the current Channel. The size cannot be smaller than the batchSize of Source. Setting the same size as batchSize is recommended.	61200
hdfs.path	Specifies the HDFS data write directory. This parameter cannot be left blank.	hdfs://hacluster/flume/test
hdfs.inUsePrefix	Specifies the prefix of the file that is being written to HDFS.	TMP_

Parameter	Description	Example Value
hdfs.batchSize	Specifies the maximum number of events that can be written to HDFS once.	61200
hdfs.kerberosPrincipal	Specifies the Kerberos authentication user, which is mandatory in security versions. This configuration is required only in security clusters.	flume_hdfs
hdfs.kerberosKeytab	Specifies the keytab file path for Kerberos authentication, which is mandatory in security versions. This configuration is required only in security clusters.	/opt/test/conf/user.keytab **NOTE** Obtain the **user.keytab** file from the Kerberos certificate file of the user **flume_hdfs**. In addition, ensure that the user who installs and runs the Flume client has the read and write permissions on the **user.keytab** file.
hdfs.useLocalTimeStamp	Specifies whether to use the local time. Possible values are **true** and **false**.	true

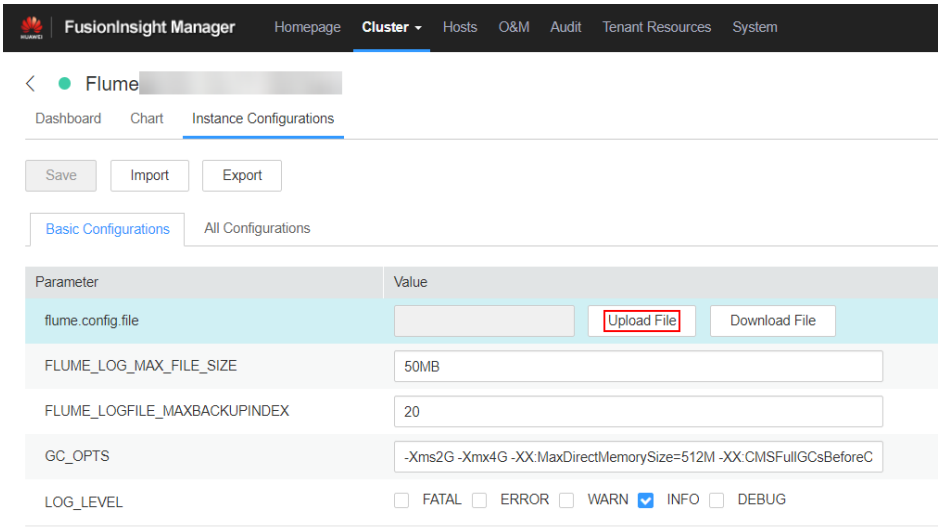
4. Log in to FusionInsight Manager and choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume**. In the **Instances** tab, click the Flume role whose configuration file is to be uploaded.

Figure 7-37 Clicking the Flume role



5. On the **Instance Configurations** page, click **Upload File** next to **flume.config.file** and select the **properties.properties** file.

Figure 7-38 Uploading a file



NOTE

- An independent server configuration file can be uploaded to each Flume instance.
- This step is required for updating the configuration file. Modifying the configuration file on the background is an improper operation because the modification will be overwritten after configuration synchronization.

6. Click **Save**, and then click **OK**.

7. Click **Finish**.

Step 4 Configure the client parameters of the Flume role.

1. Run the following commands to copy the generated client certificate (**flume_cChat.jks**) and client trust list (**flume_cChatt.jks**) to the client directory, for example, **/opt/flume-client/fusionInsight-flume-1.9.0/conf/**. (The Flume client must have been installed.) **10.196.26.1** is the service plane IP address of the node where the client resides.

   scp ${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/conf/flume_cChat.jks user@10.196.26.1:/opt/flume-client/fusionInsight-flume-1.9.0/conf/

   scp ${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/conf/flume_cChatt.jks user@10.196.26.1:/opt/flume-client/fusionInsight-flume-1.9.0/conf/

   NOTE

   When copying the client certificate, you need to enter the password of user **user** of the host (for example, **10.196.26.1**) where the client resides.

2. Log in to the node where the Flume client is decompressed as user **user**. Run the following command to go to the client directory **/opt/flume-client/fusionInsight-flume-1.9.0/bin**.

   cd opt/flume-client/fusionInsight-flume-1.9.0/bin

3. Run the following command to generate and obtain Flume client keystore password, trust list password, and keystore-password encrypted private key information. Enter the password twice and confirm the password. The password is the same as the password of the certificate whose alias is *flumechatclient* and the password of the *flume_cChat.jks* certificate library.

   ./genPwFile.sh

   cat password.property

   NOTE

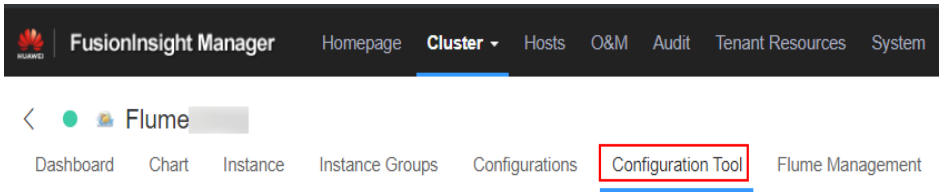
   If the following error message is displayed, run the export **JAVA_HOME=*JDKpath*** command.

   JAVA_HOME is null in current user,please install the JDK and set the JAVA_HOME

4. Run the **echo $SCC_PROFILE_DIR** command to check whether the **SCC_PROFILE_DIR** environment variable is empty.
   - If yes, run the **source .sccfile** command.
   - If no, go to **Step 4.5**.

5. Use the Flume configuration tool on FusionInsight Manager to configure the Flume role client parameters and generate a configuration file.
   a. Log in to FusionInsight Manager and choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume** > **Configuration Tool**.

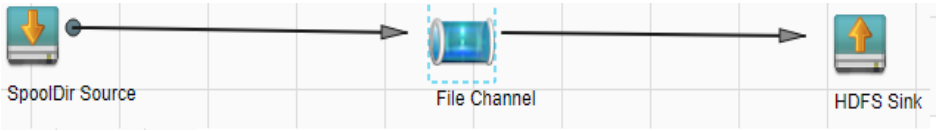
   Figure 7-39 Choosing **Configuration Tool**



b. Set **Agent Name** to **client**. Select the source, channel, and sink to be used, drag them to the GUI on the right, and connect them.

Use SpoolDir Source, File Channel, and HDFS Sink.

Figure 7-40 Example for the Flume configuration tool



c. Double-click the source, channel, and sink. Set corresponding configuration parameters by seeing **Table 7-32** based on the actual environment.

NOTE

- If the client parameters of the Flume role have been configured, you can obtain the existing client parameter configuration file from *client installation directory*/**fusioninsight-flume-1.9.0/conf/properties.properties** to ensure that the configuration is in concordance with the previous. Log in to FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume** > **Configuration Tool** > **Import**, import the file, and modify the configuration items related to encrypted transmission.
- It is recommended that the numbers of Sources, Channels, and Sinks do not exceed 40 during configuration file import. Otherwise, the response time may be very long.

d. Click **Export** to save the **properties.properties** configuration file to the local.

Table 7-32 Parameters to be modified of the Flume role client

Parameter	Description	Example Value
Name	The value must be unique and cannot be left blank.	test
spoolDir	Specifies the directory where the file to be collected resides. This parameter cannot be left blank. The directory needs to exist and have the write, read, and execute permissions on the flume running user.	/srv/BigData/hadoop/data1/zb
trackerDir	Specifies the path for storing the metadata of files collected by Flume.	/srv/BigData/hadoop/data1/tracker

Parameter	Description	Example Value
batch-size	Specifies the number of events that Flume sends in a batch.	61200
dataDirs	Specifies the directory for storing buffer data. The run directory is used by default. Configuring multiple directories on disks can improve transmission efficiency. Use commas (,) to separate multiple directories. If the directory is inside the cluster, the **/srv/BigData/hadoop/dataX/flume/data** directory can be used. **dataX** ranges from data1 to dataN. If the directory is outside the cluster, it needs to be independently planned.	/srv/BigData/hadoop/data1/flume/data
checkpointDir	Specifies the directory for storing the checkpoint information, which is under the run directory by default. If the directory is inside the cluster, the **/srv/BigData/hadoop/dataX/flume/checkpoint** directory can be used. **dataX** ranges from data1 to dataN. If the directory is outside the cluster, it needs to be independently planned.	/srv/BigData/hadoop/data1/flume/checkpoint

Parameter	Description	Example Value
transactionCapacity	Specifies the transaction size, that is, the number of events in a transaction that can be processed by the current Channel. The size cannot be smaller than the batchSize of Source. Setting the same size as batchSize is recommended.	61200
hostname	Specifies the name or IP address of the host whose data is to be sent. This parameter cannot be left blank. Name or IP address must be configured to be the name or IP address that the Avro source associated with it.	192.168.108.11
port	Specifies the IP address to which Avro Sink is bound. This parameter cannot be left blank. It must be consistent with the port that is monitored by the connected Avro Source.	21154

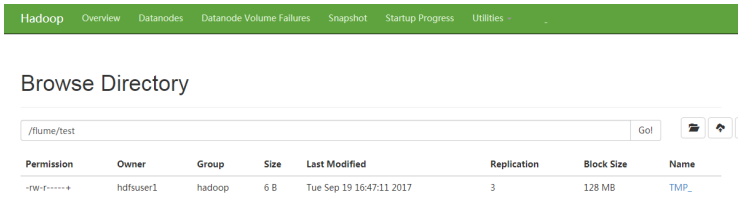
Parameter	Description	Example Value
ssl	Specifies whether to enable the SSL authentication. (You are advised to enable this function to ensure security.) Only Sources of the Avro type have this configuration item. ■ **true** indicates that the function is enabled. ■ **false** indicates that the client authentication function is not enabled.	true
keystore	Specifies the **flume_cChat.jks** certificate generated on the server.	/opt/flume-client/fusionInsight-flume-1.9.0/conf/flume_cChat.jks
keystore-password	Specifies the password of the key library, which is the password required to obtain the keystore information. Enter the value of **password** obtained in **Step 4.3**.	-
truststore	Indicates the SSL certificate trust list of the server.	/opt/flume-client/fusionInsight-flume-1.9.0/conf/flume_cChatt.jks
truststore-password	Specifies the trust list password, which is the password required to obtain the truststore information. Enter the value of **password** obtained in **Step 4.3**.	-

6. Upload the **properties.properties** file to **flume/conf/** under the installation directory of the Flume client.

Step 5 Verify log transmission.

1. Log in to FusionInsight Manager as a user who has the management permission on HDFS. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > *Name of the desired cluster* > **Services** > **HDFS**, click the HDFS WebUI link to go to the HDFS WebUI, and choose **Utilities** > **Browse the file system**.
2. Check whether the data is generated in the **/flume/test** directory on the HDFS.

Figure 7-41 Checking HDFS directories and files



----End

7.7 Enterprise-Class Enhancements of Flume

7.7.1 Using the Encryption Tool of the Flume Client

You can use the encryption tool provided by the Flume client to encrypt some parameter values in the configuration file.

Step 1 Install the Flume client.

Step 2 Log in to the Flume client node and go to the client installation directory, for example, **/opt/FlumeClient**.

Step 3 Run the following command to switch the directory:

cd fusioninsight-flume-*Flume component version number***/bin**

Step 4 Run the following command to encrypt information:

./genPwFile.sh

Input the information that you want to encrypt twice.

Step 5 Run the following command to query the encrypted information:

cat password.property

NOTE

- If the encryption parameter is used for the Flume server, you need to perform encryption on the corresponding Flume server node. You need to run the encryption script as user **omm** for encryption.

----End

7.7.2 Configuring Flume to Connect to Kafka in Security Mode

This section describes how to connect to Kafka using the Flume client in security mode.

Step 1 Create a **jaas.conf** file and save it to **${***Flume client installation directory***} /conf**. The content of the **jaas.conf** file is as follows:

```
KafkaClient {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
keyTab="/opt/test/conf/user.keytab"
principal="flume_hdfs@<System domain name>"
useTicketCache=false
storeKey=true
debug=true;
 };
```

Set **keyTab** and **principal** based on site requirements. The configured **principal** must have certain kafka permissions.

Step 2 Configure services. Set the port number of **kafka.bootstrap.servers** to **21007**, and set **kafka.security.protocol** to **SASL_PLAINTEXT**.

Step 3 If the domain name of the cluster where Kafka is located is changed, change the value of *-Dkerberos.domain.name* in the **flume-env.sh** file in **${***Flume client installation directory***} /conf/** based on the site requirements.

Step 4 Upload the configured **properties.properties** file to **${***Flume client installation directory***} /conf**.

----End

7.8 Flume O&M Management

7.8.1 Flume Common Configuration Parameters

Some parameters can be configured on Manager.

This section describes how to configure the sources, channels, and sinks of Flume, and modify the configuration items of each module.

For MRS 3.*x* or later, log in to FusionInsight Manager and choose **Cluster** > **Services** > **Flume**. On the displayed page, click the **Configuration Tool** tab, select and drag the source, channel, and sink to be used to the GUI on the right, and double-click them to configure corresponding parameters. Parameters such as **channels** and **type** are configured only in the client configuration file **properties.properties**, the path of which is *Flume client installation directory*/**fusioninsight-flume-***Flume version*/**conf/properties.properties**.

NOTE

You must input encrypted information for some configurations. For details on how to encrypt information, see **Using the Encryption Tool of the Flume Client**.

Common Source Configurations

- **Avro Source**

  An Avro source listens to the Avro port, receives data from the external Avro client, and places data into configured channels. **Table 7-33** lists common configurations.

  Figure 7-42 Avro Source

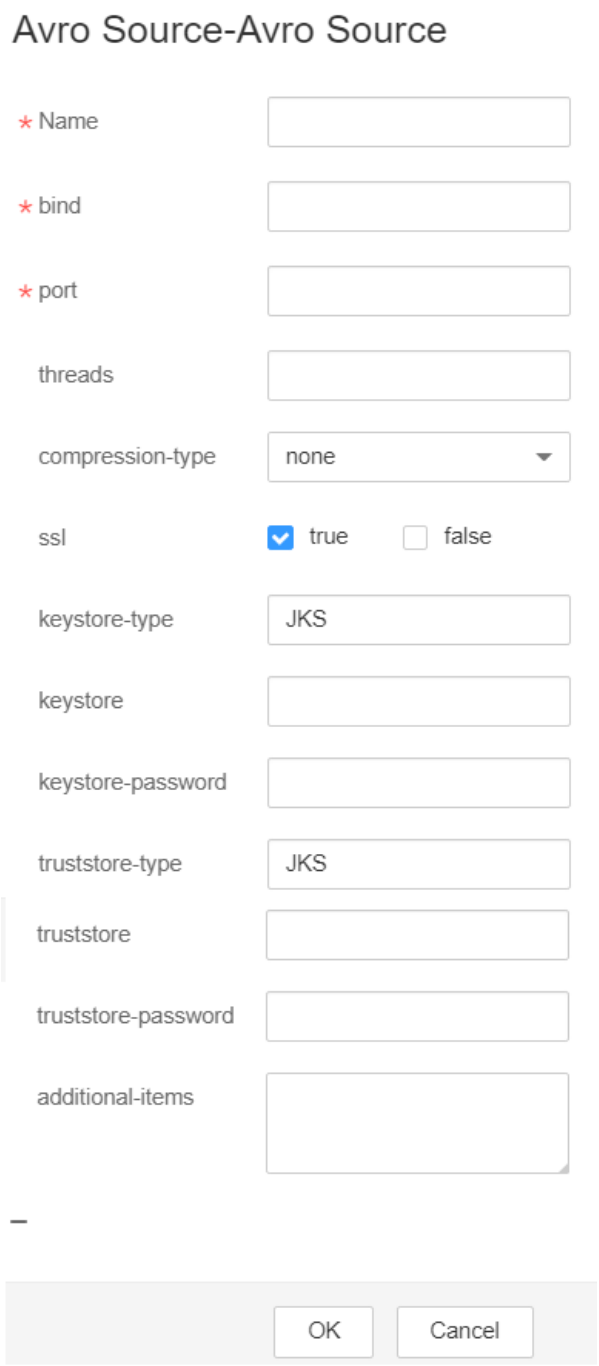


Table 7-33 Common configurations of an Avro source

Parameter	Default Value	Description
channels	-	Specifies the channel connected to the source. Multiple channels can be configured. Use spaces to separate them. In a single proxy process, sources and sinks are connected through channels. A source instance corresponds to multiple channels, but a sink instance corresponds only to one channel. The format is as follows: **<Agent >.sources.<Source>.channels = <channel1> <channel2> <channel3>...** **<Agent >.sinks.<Sink>.channels = <channel1>** This parameter can be configured only in the **properties.properties** file.
type	avro	Specifies the type, which is set to **avro**. The type of each source is a fixed value. This parameter can be configured only in the **properties.properties** file.
bind	-	Specifies the host name or IP address associated with the source.
port	-	Specifies the bound port number.
ssl	false	Specifies whether to use SSL encryption. • true • false
truststore-type	JKS	Specifies the Java trust store type. Set this parameter to **JKS** or other truststore types supported by Java.
truststore	-	Specifies the Java trust store file.
truststore-password	-	Specifies the Java trust store password.

Parameter	Default Value	Description
keystore-type	JKS	Specifies the key storage type. Set this parameter to **JKS** or other truststore types supported by Java.
keystore	-	Specifies the key storage file.
keystore-password	-	Specifies the key storage password.

- **SpoolDir Source**

  A SpoolDir source monitors and transmits new files that have been added to directories in quasi-real-time mode. Common configurations are as follows:

Figure 7-43 SpoolDir Source

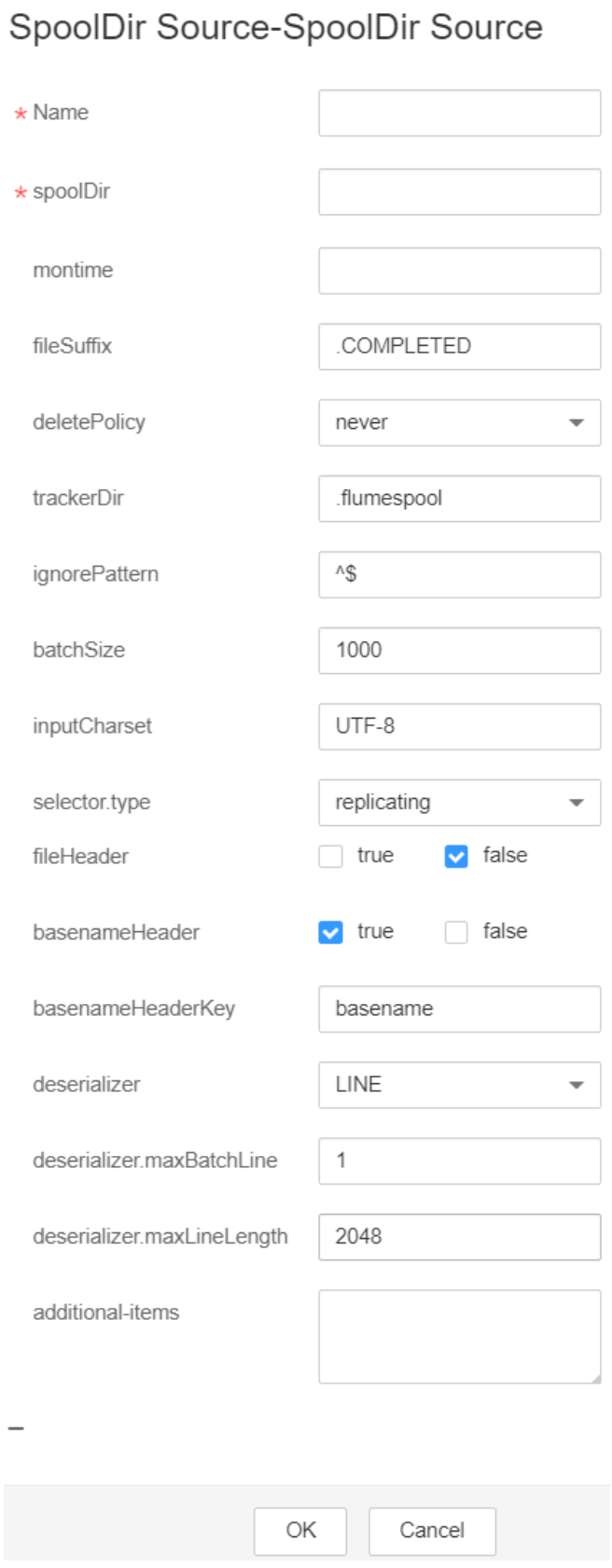


Table 7-34 Common configurations of a SpoolDir source

Parameter	Default Value	Description
channels	-	Specifies the channel connected to the source. Multiple channels can be configured. This parameter can be configured only in the **properties.properties** file.
type	spooldir	Type, which is set to **spooldir**. This parameter can be configured only in the **properties.properties** file.
monTime	0 (Disabled)	Specifies the thread monitoring threshold. When the update time exceeds the threshold, the source is restarted. Unit: second
spoolDir	-	Specifies the monitoring directory.
fileSuffix	.COMPLETED	Specifies the suffix added after file transmission is complete.
deletePolicy	never	Specifies the source file deletion policy after file transmission is complete. The value can be either **never** or **immediate**.
ignorePattern	^$	Specifies the regular expression of a file to be ignored.
trackerDir	.flumespool	Specifies the metadata storage path during data transmission.
batchSize	1000	Specifies the source transmission granularity.

Parameter	Default Value	Description
decodeErrorPolicy	FAIL	Specifies the code error policy. This parameter can be configured only in the **properties.properties** file. The value can be **FAIL**, **REPLACE**, or **IGNORE**. **FAIL**: Generate an exception and fail the parsing. **REPLACE**: Replace the characters that cannot be identified with other characters, such as U+FFFD. **IGNORE**: Discard character strings that cannot be parsed. **NOTE** If a code error occurs in the file, set **decodeErrorPolicy** to **REPLACE** or **IGNORE**. Flume will skip the code error and continue to collect subsequent logs.
deserializer	LINE	Specifies the file parser. The value can be either **LINE** or **BufferedLine**. ● When the value is set to **LINE**, characters read from the file are transcoded one by one. ● When the value is set to **BufferedLine**, one line or multiple lines of characters read from the file are transcoded in batches, which delivers better performance.
deserializer.maxLineLength	2048	Specifies the maximum length for resolution by line, ranging from 0 to 2,147,483,647.
deserializer.maxBatchLine	1	Specifies the maximum number of lines for resolution by line. If multiple lines are set, **maxLineLength** must be set to a corresponding multiplier. For example, if **maxBatchLine** is set to **2**, **maxLineLength** is set to **4096** (2048 x 2).

Parameter	Default Value	Description
selector.type	replicating	Specifies the selector type. The value can be either **replicating** or **multiplexing**. • **replicating** indicates that the same content is sent to each channel. • **multiplexing** indicates that the content is sent only to certain channels according to the distribution rule.
interceptors	-	Specifies the interceptor. This parameter can be configured only in the **properties.properties** file.

NOTE

The Spooling source ignores the last line feed character of each event when data is read by line. Therefore, Flume does not calculate the data volume counters used by the last line feed character.

- **Kafka Source**

  A Kafka source consumes data from Kafka topics. Multiple sources can consume data of the same topic, and the sources consume different partitions of the topic. Common configurations are as follows:

Figure 7-44 Kafka Source

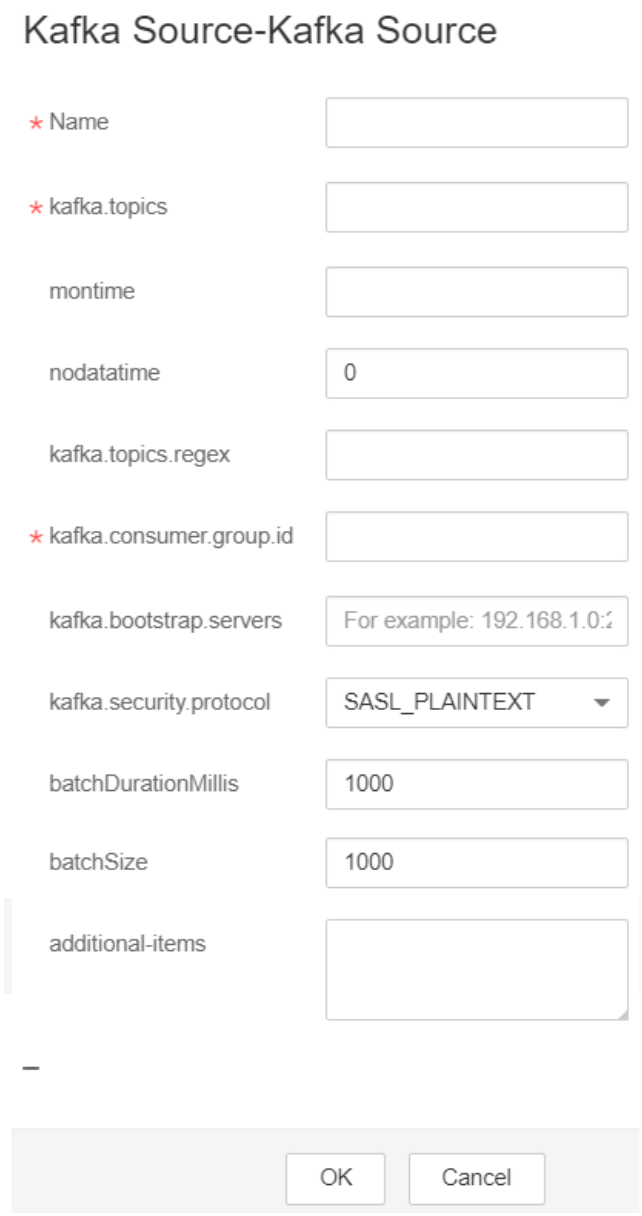


Table 7-35 Common configurations of a Kafka source

Parameter	Default Value	Description
channels	-	Specifies the channel connected to the source. Multiple channels can be configured. This parameter can be configured only in the **properties.properties** file.

Parameter	Default Value	Description
type	org.apache.flume.source.kafka.KafkaSource	Specifies the type, which is set to **org.apache.flume.source.kafka.KafkaSource**. This parameter can be configured only in the **properties.properties** file.
monTime	0 (Disabled)	Specifies the thread monitoring threshold. When the update time exceeds the threshold, the source is restarted. Unit: second
nodatatime	0 (Disabled)	Specifies the alarm threshold. An alarm is triggered when the duration that Kafka does not release data to subscribers exceeds the threshold. Unit: second
batchSize	1000	Specifies the number of events written into a channel at a time.
batchDurationMillis	1000	Specifies the maximum duration of topic data consumption at a time, expressed in milliseconds.
keepTopicInHeader	false	Specifies whether to save topics in the event header. If topics are saved, topics configured in Kafka sinks become invalid. ● true ● false This parameter can be configured only in the **properties.properties** file.
keepPartitionIn-Header	false	Specifies whether to save partition IDs in the event header. If partition IDs are saved, Kafka sinks write data to the corresponding partitions. ● true ● false This parameter can be set only in the properties.properties file.
kafka.bootstrap.servers	-	Specifies the list of Broker addresses, which are separated by commas.
kafka.consumer.group.id	-	Specifies the Kafka consumer group ID.

Parameter	Default Value	Description
kafka.topics	-	Specifies the list of subscribed Kafka topics, which are separated by commas (,).
kafka.topics.regex	-	Specifies the subscribed topics that comply with regular expressions. **kafka.topics.regex** has a higher priority than **kafka.topics** and will overwrite **kafka.topics**.
kafka.security.protocol	SASL_PLAINTEXT	Specifies the security protocol of Kafka. The value must be set to **PLAINTEXT** for clusters in which Kerberos authentication is disabled.
kafka.kerberos.domain.name	-	Specifies the value of **default_realm** of Kerberos in the Kafka cluster, which should be configured only for security clusters. This parameter can be set only in the properties.properties file.
Other Kafka Consumer Properties	-	Specifies other Kafka configurations. This parameter can be set to any consumption configuration supported by Kafka, and the **.kafka** prefix must be added to the configuration. This parameter can be set only in the properties.properties file.

- **Taildir Source**

  A Taildir source monitors file changes in a directory and automatically reads the file content. In addition, it can transmit data in real time. **Table 7-36** lists common configurations.

Figure 7-45 Taildir Source

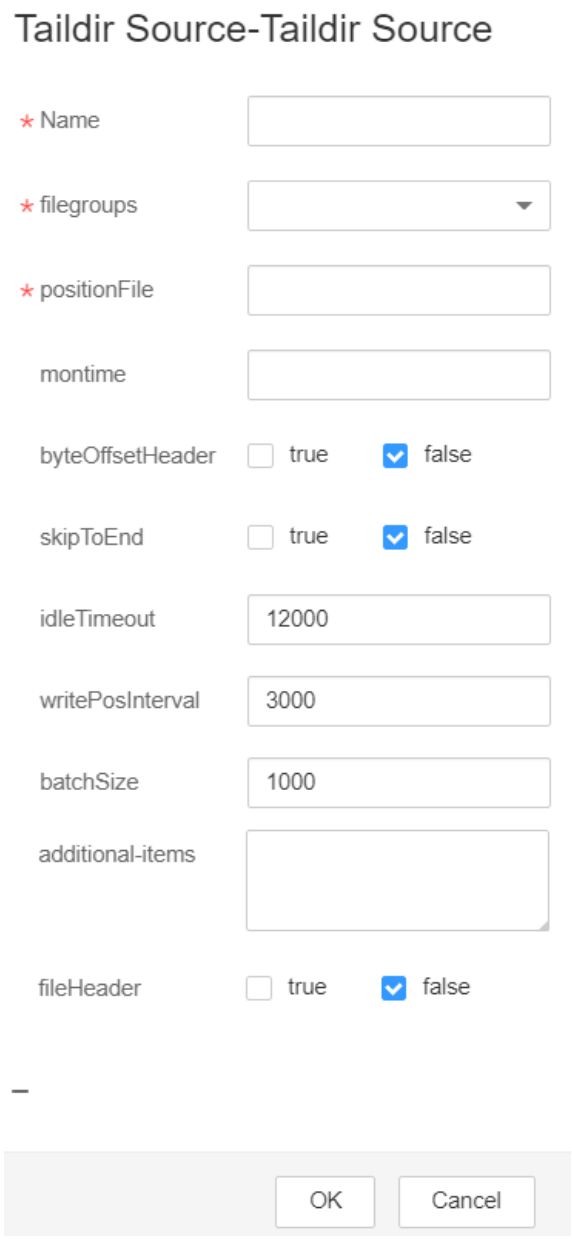


Table 7-36 Common configurations of a Taildir source

Parameter	Default Value	Description
channels	-	Specifies the channel connected to the source. Multiple channels can be configured. This parameter can be set only in the properties.properties file.

Parameter	Default Value	Description
type	taildir	Specifies the type, which is set to **taildir**. This parameter can be set only in the properties.properties file.
filegroups	-	Specifies the group name of a collection file directory. Group names are separated by spaces.
filegroups.<filegroupName>.parentDir	-	Specifies the parent directory. The value must be an absolute path. This parameter can be set only in the properties.properties file.
filegroups.<filegroupName>.filePattern	-	Specifies the relative file path of the file group's parent directory. Directories can be included and regular expressions are supported. It must be used together with **parentDir**. This parameter can be set only in the properties.properties file.
positionFile	-	Specifies the metadata storage path during data transmission.
headers.<filegroupName>.<headerKey>	-	Specifies the key-value of an event when data of a group is being collected. This parameter can be set only in the properties.properties file.
byteOffsetHeader	false	Specifies whether each event header should contain the location information about the event in the source file. The location information is saved in the **byteoffset** variable.
skipToEnd	false	Specifies whether Flume can locate the latest location of a file and read the latest data after restart.
idleTimeout	120000	Specifies the idle duration during file reading, expressed in milliseconds. If the file data is not changed in this idle period, the source closes the file. If data is written into this file after it is closed, the source opens the file and reads data.

Parameter	Default Value	Description
writePosInterval	3000	Specifies the interval for writing metadata to a file, expressed in milliseconds.
batchSize	1000	Specifies the number of events written to the channel in batches.
monTime	0 (Disabled)	Specifies the thread monitoring threshold. When the update time exceeds the threshold, the source is restarted. Unit: second

- **Http Source**

  An HTTP source receives data from an external HTTP client and sends the data to the configured channels. **Table 7-37** lists common configurations.

  Figure 7-46 Http Source

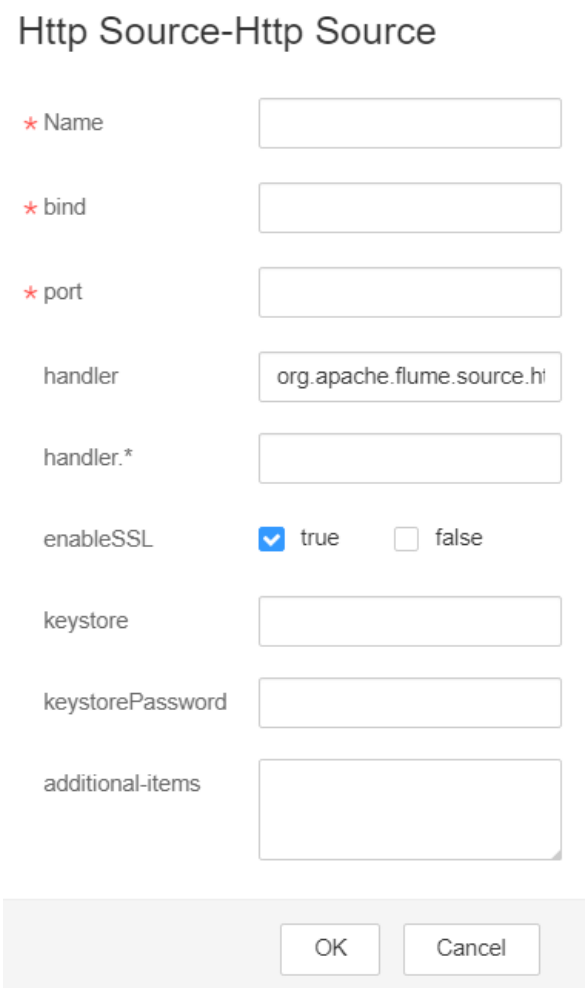


Table 7-37 Common configurations of an HTTP source

Parameter	Default Value	Description
channels	-	Specifies the channel connected to the source. Multiple channels can be configured. This parameter can be set only in the properties.properties file.
type	http	Specifies the type, which is set to **http**. This parameter can be set only in the properties.properties file.
bind	-	Specifies the name or IP address of the bound host.
port	-	Specifies the bound port.
handler	org.apache.flume.source.http.JSONHandler	Specifies the message parsing method of an HTTP request. The following methods are supported: • **org.apache.flume.source.http.JSONHandler**: JSON • **org.apache.flume.sink.solr.morphline.BlobHandler**: BLOB
handler.*	-	Specifies handler parameters.
enableSSL	false	Specifies whether SSL is enabled in HTTP.
keystore	-	Specifies the keystore path set after SSL is enabled in HTTP.
keystorePassword	-	Specifies the keystore password set after SSL is enabled in HTTP.

Common Channel Configurations

- **Memory Channel**

  A memory channel uses memory as the cache. Events are stored in memory queues. **Table 7-38** lists common configurations.

Figure 7-47 Memory Channel

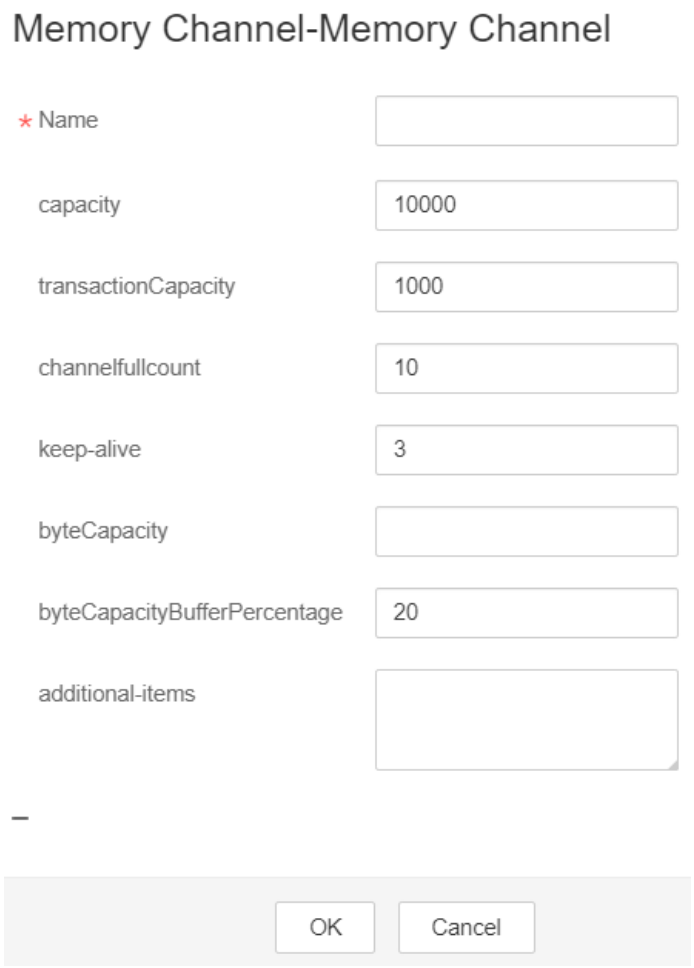


Table 7-38 Common configurations of a memory channel

Parameter	Default Value	Description
type	-	Specifies the type, which is set to **memory**. This parameter can be set only in the properties.properties file.
capacity	10000	Specifies the maximum number of events cached in a channel.
transactionCapacity	1000	Specifies the maximum number of events accessed each time.
channelfullcount	10	Specifies the channel full count. When the count reaches the threshold, an alarm is reported.

- **File Channel**

  A file channel uses local disks as the cache. Events are stored in the folder specified by **dataDirs**. Table 7-39 lists common configurations.

Figure 7-48 File Channel

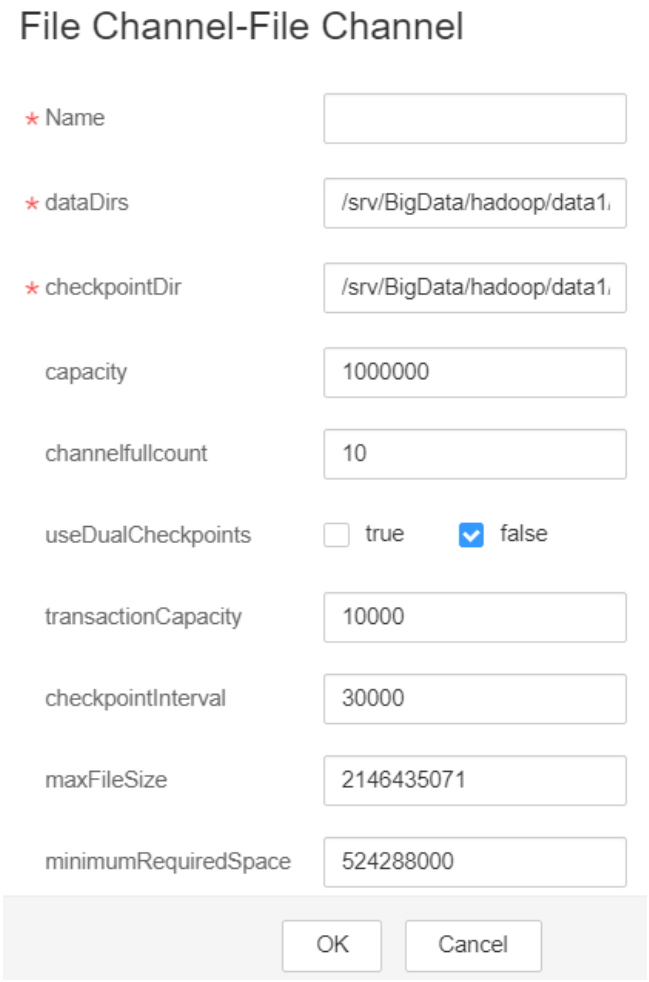


Table 7-39 Common configurations of a file channel

Parameter	Default Value	Description
type	-	Specifies the type, which is set to **file**. This parameter can be set only in the properties.properties file.
checkpointDir	${BIGDATA_DATA_HOME}/flume/checkpoint	Specifies the checkpoint storage directory.
dataDirs	${BIGDATA_DATA_HOME}/flume/data	Specifies the data cache directory. Multiple directories can be configured to improve performance. The directories are separated by commas (,).

Parameter	Default Value	Description
maxFileSize	2146435071	Specifies the maximum size of a single cache file, expressed in bytes.
minimumRequired-Space	524288000	Specifies the minimum idle space in the cache, expressed in bytes.
capacity	1000000	Specifies the maximum number of events cached in a channel.
transactionCapacity	10000	Specifies the maximum number of events accessed each time.
channelfullcount	10	Specifies the channel full count. When the count reaches the threshold, an alarm is reported.

- **Kafka Channel**

  A Kafka channel uses a Kafka cluster as the cache. Kafka provides high availability and multiple copies to prevent data from being immediately consumed by sinks when Flume or Kafka Broker crashes. **Table 10 Common configurations of a Kafka channel** lists common configurations.

Figure 7-49 Kafka Channel

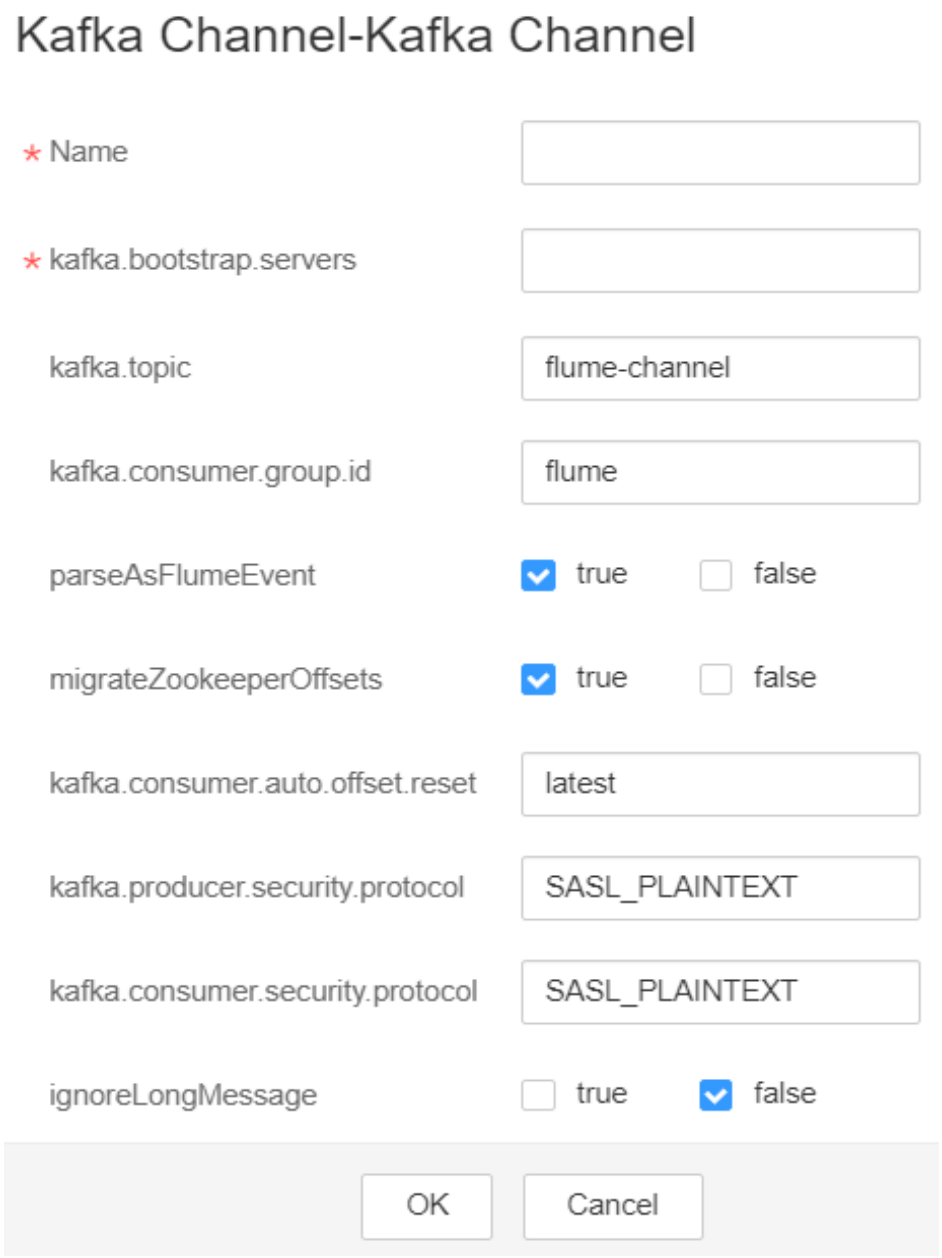


Table 7-40 Common configurations of a Kafka channel

Parameter	Default Value	Description
type	-	Specifies the type, which is set to **org.apache.flume.channel.kafka.KafkaChannel**. This parameter can be set only in the properties.properties file.
kafka.bootstrap.servers	-	Specifies the list of Brokers in the Kafka cluster.

Parameter	Default Value	Description
kafka.topic	flume-channel	Specifies the Kafka topic used by the channel to cache data.
kafka.consumer.group.id	flume	Specifies the Kafka consumer group ID.
parseAsFlumeEvent	true	Specifies whether data is parsed into Flume events.
migrateZookeeper-Offsets	true	Specifies whether to search for offsets in ZooKeeper and submit them to Kafka when there is no offset in Kafka.
kafka.consumer.auto.offset.reset	latest	Consumes data from the specified location when there is no offset.
kafka.producer.security.protocol	SASL_PLAINTEXT	Specifies the Kafka producer security protocol.
kafka.consumer.security.protocol	SASL_PLAINTEXT	Specifies the Kafka consumer security protocol.

Common Sink Configurations

- **HDFS Sink**

  An HDFS sink writes data into HDFS. **Table 7-41** lists common configurations.

Figure 7-50 HDFS Sink

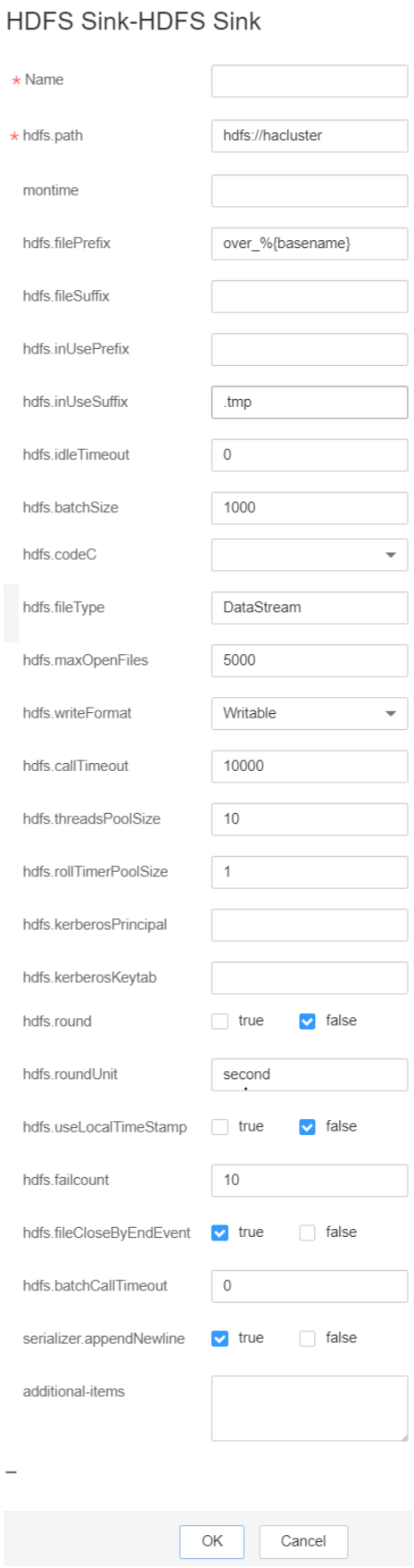


Table 7-41 Common configurations of an HDFS sink

Parameter	Default Value	Description
channel	-	Specifies the channel connected to the sink. This parameter can be set only in the properties.properties file.
type	hdfs	Specifies the type, which is set to **hdfs**. This parameter can be set only in the properties.properties file.
monTime	0 (Disabled)	Specifies the thread monitoring threshold. When the update time exceeds the threshold, the sink is restarted. Unit: second
hdfs.path	-	Specifies the HDFS path.
hdfs.inUseSuffix	.tmp	Specifies the suffix of the HDFS file to which data is being written.
hdfs.rollInterval	30	Specifies the interval for file rolling, expressed in seconds. Set **hdfs.fileCloseByEndEvent** to **false** if you set this parameter.
hdfs.rollSize	1024	Specifies the size for file rolling, expressed in bytes. Set **hdfs.fileCloseByEndEvent** to **false** if you set this parameter.
hdfs.rollCount	10	Specifies the number of events for file rolling. Set **hdfs.fileCloseByEndEvent** to **false** if you set this parameter.
hdfs.idleTimeout	0	Specifies the timeout interval for closing idle files automatically, expressed in seconds.
hdfs.batchSize	1000	Specifies the number of events written into HDFS at a time.
hdfs.kerberosPrincipal	-	Specifies the Kerberos username for HDFS authentication. This parameter is not required for a cluster in which Kerberos authentication is disabled.
hdfs.kerberosKeytab	-	Specifies the Kerberos keytab of HDFS authentication. This parameter is not required for a cluster in which Kerberos authentication is disabled.

Parameter	Default Value	Description
hdfs.fileCloseByEndEvent	true	Specifies whether to close the file when the last event is received.
hdfs.batchCallTimeout	-	Specifies the timeout control duration each time events are written into HDFS, expressed in milliseconds. If this parameter is not specified, the timeout duration is controlled when each event is written into HDFS. When the value of **hdfs.batchSize** is greater than 0, configure this parameter to improve the performance of writing data into HDFS. **NOTE** The value of **hdfs.batchCallTimeout** depends on **hdfs.batchSize**. A greater **hdfs.batchSize** requires a larger **hdfs.batchCallTimeout**. If the value of **hdfs.batchCallTimeout** is too small, writing events to HDFS may fail.
serializer.appendNewline	true	Specifies whether to add a line feed character (**\n**) after an event is written to HDFS. If a line feed character is added, the data volume counters used by the line feed character will not be calculated by HDFS sinks.

- **Avro Sink**

  An Avro sink converts events into Avro events and sends them to the monitoring ports of the hosts. **Table 7-42** lists common configurations.

Figure 7-51 Avro Sink

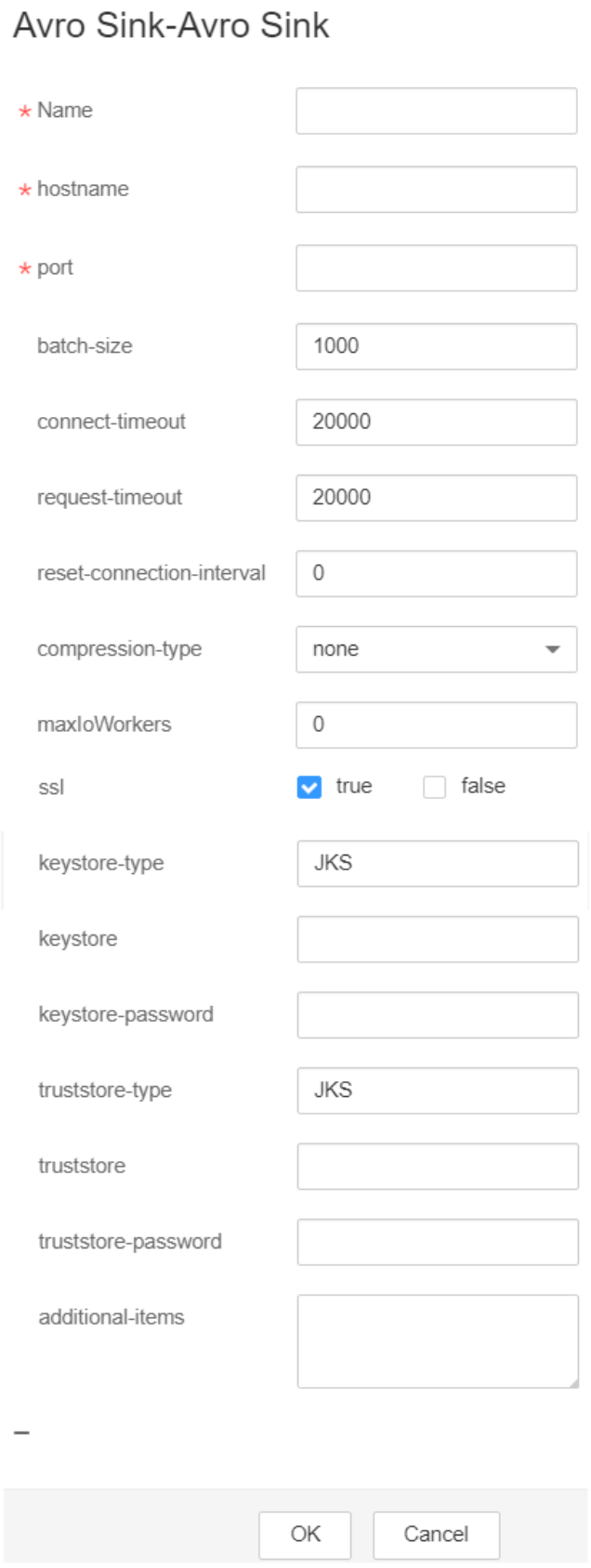


Table 7-42 Common configurations of an Avro sink

Parameter	Default Value	Description
channel	**-**	Specifies the channel connected to the sink. This parameter can be set only in the properties.properties file.
type	-	Specifies the type, which is set to **avro**. This parameter can be set only in the properties.properties file.
hostname	-	Specifies the name or IP address of the bound host.
port	-	Specifies the monitoring port.
batch-size	1000	Specifies the number of events sent in a batch.
ssl	false	Specifies whether to use SSL encryption.
truststore-type	JKS	Specifies the Java trust store type.
truststore	-	Specifies the Java trust store file.
truststore-password	-	Specifies the Java trust store password.
keystore-type	JKS	Specifies the key storage type.
keystore	-	Specifies the key storage file.
keystore-password	-	Specifies the key storage password.

- **HBase Sink**

  An HBase sink writes data into HBase. **Table 7-43** lists common configurations.

Figure 7-52 HBase Sink

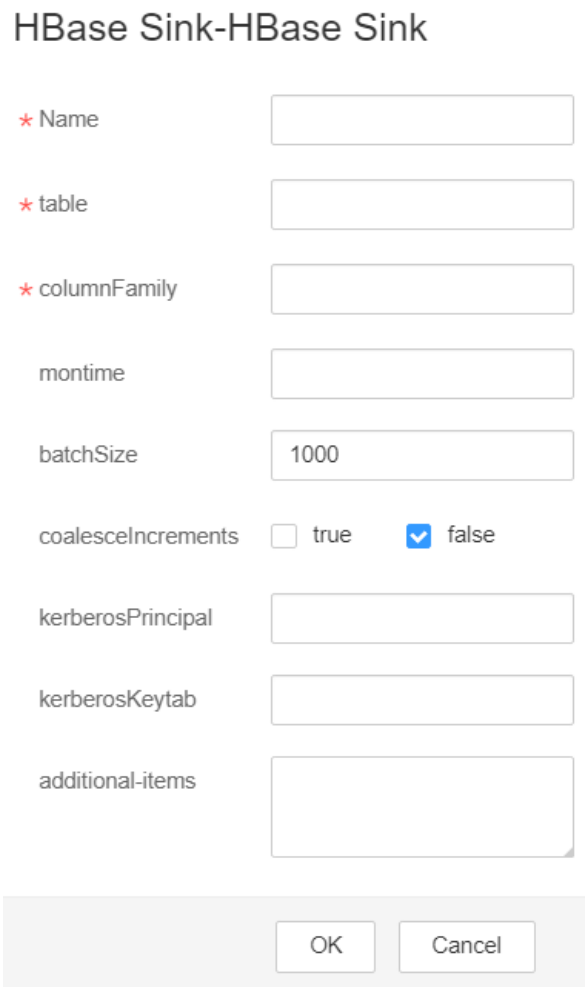


Table 7-43 Common configurations of an HBase sink

Parameter	Default Value	Description
channel	-	Specifies the channel connected to the sink. This parameter can be set only in the properties.properties file.
type	-	Specifies the type, which is set to **hbase**. This parameter can be set only in the properties.properties file.
table	-	Specifies the HBase table name.
monTime	0 (Disabled)	Specifies the thread monitoring threshold. When the update time exceeds the threshold, the sink is restarted. Unit: second
columnFamily	-	Specifies the HBase column family.

Parameter	Default Value	Description
batchSize	1000	Specifies the number of events written into HBase at a time.
kerberosPrincipal	-	Specifies the Kerberos username for HBase authentication. This parameter is not required for a cluster in which Kerberos authentication is disabled.
kerberosKeytab	-	Specifies the Kerberos keytab of HBase authentication. This parameter is not required for a cluster in which Kerberos authentication is disabled.

- **Kafka Sink**

  A Kafka sink writes data into Kafka. **Table 7-44** lists common configurations.

  Figure 7-53 Kafka Sink

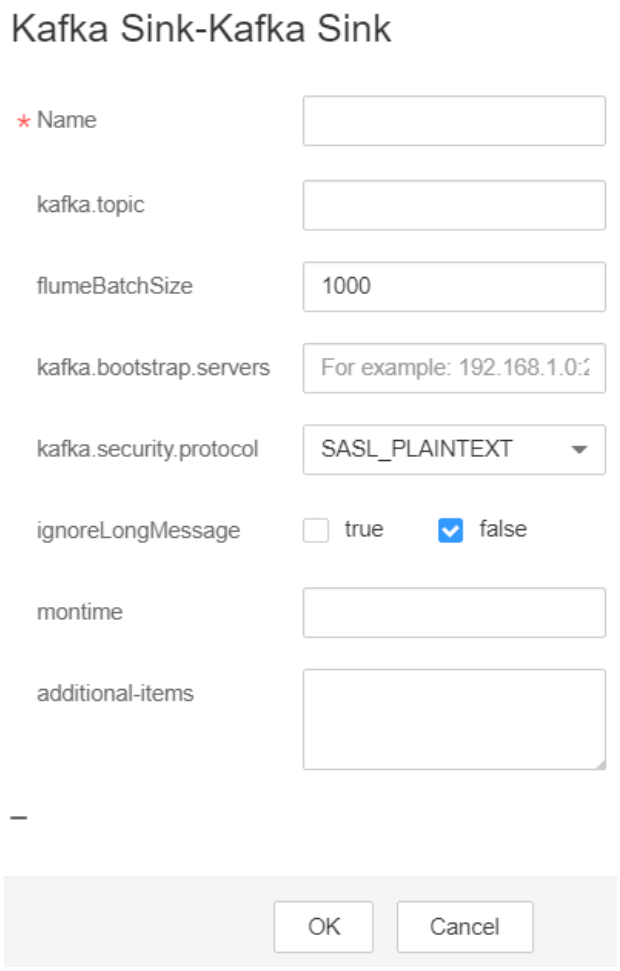


Table 7-44 Common configurations of a Kafka sink

Parameter	Default Value	Description
channel	**-**	Specifies the channel connected to the sink. This parameter can be set only in the properties.properties file.
type	-	Specifies the type, which is set to **org.apache.flume.sink.kafka.KafkaSink**. This parameter can be set only in the properties.properties file.
kafka.bootstrap.servers	-	Specifies the list of Kafka Brokers, which are separated by commas.
monTime	0 (Disabled)	Specifies the thread monitoring threshold. When the update time exceeds the threshold, the sink is restarted. Unit: second
kafka.topic	default-flume-topic	Specifies the topic where data is written.
flumeBatchSize	1000	Specifies the number of events written into Kafka at a time.
kafka.security.protocol	SASL_PLAINTEXT	Specifies the security protocol of Kafka. The value must be set to **PLAINTEXT** for clusters in which Kerberos authentication is disabled.
kafka.kerberos.domain.name	-	Specifies the Kafka domain name. This parameter is mandatory for a security cluster. This parameter can be set only in the properties.properties file.
Other Kafka Producer Properties	-	Specifies other Kafka configurations. This parameter can be set to any production configuration supported by Kafka, and the **.kafka** prefix must be added to the configuration. This parameter can be set only in the properties.properties file.

7.8.2 Flume Log Overview

Log Description

Log path: The default path of Flume log files is **/var/log/Bigdata/***Role name*.

- FlumeServer: **/var/log/Bigdata/flume/flume**
- FlumeClient: **/var/log/Bigdata/flume-client-n/flume**
- MonitorServer: **/var/log/Bigdata/flume/monitor**

Log archive rule: The automatic Flume log compression function is enabled. By default, when the size of logs exceeds 50 MB , logs are automatically compressed into a log file named in the following format: *<Original log file name>-<yyyy-mm-dd_hh-mm-ss>.[ID]***.log.zip**. A maximum of 20 latest compressed files are reserved. The number of compressed files can be configured on the Manager portal.

Table 7-45 Flume log list

Type	Name	Description
Run logs	/flume/flumeServer.log	Log file that records FlumeServer running environment information.
	/flume/install.log	FlumeServer installation log file
	/flume/flumeServer-gc.log.*<No.>*	GC archive log file of the FlumeServer process
	/flume/prestartDvietail.log	Work log file before the FlumeServer startup
	/flume/startDetail.log	Startup log file of the Flume process
	/flume/stopDetail.log	Shutdown log file of the Flume process
	/monitor/monitorServer.log	Log file that records MonitorServer running environment information
	/monitor/startDetail.log	Startup log file of the MonitorServer process
	/monitor/stopDetail.log	Shutdown log file of the MonitorServer process
	function.log	External function invoking log file
	/flume/flume-*Username*-*Date*-*pid*-gc.log	GC log file of the Flume process
	/flume/Flume-audit.log	Audit log file of the Flume client
	/flume/startAgent.out	Process parameter log file generated before Flume startup

Type	Name	Description
Stack information log (MRS 3.2.0 or later)	threadDump-<DATE>.log	The jstack log file to be printed when the NodeAgent delivers a service stop command

Log Level

Table 7-46 describes the log levels supported by Flume.

Levels of run logs are FATAL, ERROR, WARN, INFO, and DEBUG from the highest to the lowest priority. Run logs of equal or higher levels are recorded. The higher the specified log level, the fewer the logs recorded.

Table 7-46 Log level

Type	Level	Description
Run log	FATAL	Logs of this level record critical error information about system running.
	ERROR	Logs of this level record error information about system running.
	WARN	Logs of this level record exception information about the current event processing.
	INFO	Logs of this level record normal running status information about the system and events.
	DEBUG	Logs of this level record the system information and system debugging information.

To modify log levels, perform the following operations:

Step 1 Go to the **All Configurations** page of Flume by referring to **Modifying Cluster Service Configuration Parameters**.

Step 2 On the menu bar on the left, select the log menu of the target role.

Step 3 Select a desired log level.

Step 4 Save the configuration. In the displayed dialog box, click **OK** to make the configurations take effect.

----End

NOTE

The configurations take effect immediately without the need to restart the service.

Log Format

The following table lists the Flume log formats.

Table 7-47 Log format

Type	Format	Example
Run logs	<*yyyy-MM-dd HH:mm:ss,SSS*>|<*Log level*>|<*Name of the thread that generates the log*>|<*Message in the log*>|<*Location where the log event occurs*>	2014-12-12 11:54:57,316 | INFO | [main] | log4j dynamic load is start. | org.apache.flume.tools.LogDynamicLoad.start(LogDynamicLoad.java:59)
	<*yyyy-MM-dd HH:mm:ss,SSS*><*Username*><*User IP*><*Time*><*Operation*><*Resource*><*Result*><*Detail*>	2014-12-12 23:04:16,572 | INFO | [SinkRunner-PollingRunner-DefaultSinkProcessor] | SRCIP=null OPERATION=close

7.8.3 Viewing Flume Client Logs

Step 1 Install the Flume client.

Step 2 Go to the Flume client log directory (**/var/log/Bigdata** by default).

Step 3 Run the following command to view the log file:

ls -lR flume-client-*

A log file is shown as follows:

```
flume-client-1/flume:
total 7672
-rw-------. 1 root root       0 Sep  8 19:43 Flume-audit.log
-rw-------. 1 root root 1562037 Sep 11 06:05 FlumeClient.2017-09-11_04-05-09.[1].log.zip
-rw-------. 1 root root 6127274 Sep 11 14:47 FlumeClient.log
-rw-------. 1 root root    2935 Sep  8 22:20 flume-root-20170908202009-pid72456-gc.log.0.current
-rw-------. 1 root root    2935 Sep  8 22:27 flume-root-20170908202634-pid78789-gc.log.0.current
-rw-------. 1 root root    4382 Sep  8 22:47 flume-root-20170908203137-pid84925-gc.log.0.current
-rw-------. 1 root root    4390 Sep  8 23:46 flume-root-20170908204918-pid103920-gc.log.0.current
-rw-------. 1 root root    3196 Sep  9 10:12 flume-root-20170908215351-pid44372-gc.log.0.current
-rw-------. 1 root root    2935 Sep  9 10:13 flume-root-20170909101233-pid55119-gc.log.0.current
-rw-------. 1 root root    6441 Sep  9 11:10 flume-root-20170909101631-pid59301-gc.log.0.current
-rw-------. 1 root root       0 Sep  9 11:10 flume-root-20170909111009-pid119477-gc.log.0.current
-rw-------. 1 root root   92896 Sep 11 13:24 flume-root-20170909111126-pid120689-gc.log.0.current
-rw-------. 1 root root    5588 Sep 11 14:46 flume-root-20170911132445-pid42259-gc.log.0.current
-rw-------. 1 root root    2576 Sep 11 13:24 prestartDetail.log
```



```
-rw-------. 1 root root    3303 Sep 11 13:24 startDetail.log
-rw-------. 1 root root    1253 Sep 11 13:24 stopDetail.log

flume-client-1/monitor:
total 8
-rw-------. 1 root root  141 Sep  8 19:43 flumeMonitorChecker.log
-rw-------. 1 root root 2946 Sep 11 13:24 flumeMonitor.log
```

In the log file, **FlumeClient.log** is the run log of the Flume client.

----End

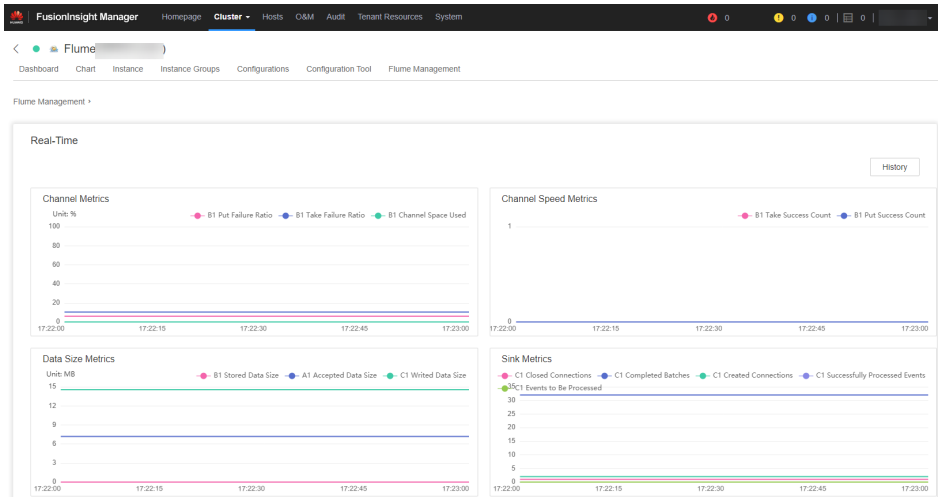
7.8.4 Viewing Flume Client Monitoring Information

The Flume client outside the FusionInsight cluster is a part of the end-to-end data collection. Both the Flume client outside the cluster and the Flume server in the cluster need to be monitored. Users can use FusionInsight Manager to monitor the Flume client and view the monitoring indicators of the Source, Sink, and Channel of the client as well as the client process status.

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume** > **Flume Management** to view the current Flume client list and process status.

Figure 7-54 Viewing Flume client monitoring information on the **Flume Management** page



Step 3 Click the **Instance ID**, and view client monitoring metrics in the **Current** area.

Step 4 Click **History**. The page for querying historical monitoring data is displayed. Select a time range and click **View** to view the monitoring data within the time range.

----End

7.8.5 Stopping or Uninstalling the Flume Client

Scenario

You can stop and start the Flume client or uninstall the Flume client when the Flume data ingestion channel is not required.

Procedure

- Stop the Flume client of the Flume role.

  Assume that the Flume client installation path is **/opt/FlumeClient**. Run the following command to stop the Flume client:

  cd /opt/FlumeClient/fusioninsight-flume-*Flume component version number***/bin**

  ./flume-manage.sh stop

  If the following information is displayed after the command execution, the Flume client is successfully stopped.

  ```
  Stop Flume PID=120689 successful..
  ```

  NOTE

  The Flume client will be automatically restarted after being stopped. If you do not need automatic restart, run the following command:

  ./flume-manage.sh stop force

  If you want to restart the Flume client, run the following command:

  ./flume-manage.sh start force

- Uninstall the Flume client of the Flume role.

  Assume that the Flume client installation path is **/opt/FlumeClient**. Run the following command to uninstall the Flume client:

  cd /opt/FlumeClient/fusioninsight-flume-*Flume component version number***/inst**

  ./uninstall.sh

7.9 Common Issues About Flume

7.9.1 How Do I View Flume Logs

Flume logs are stored in **/var/log/Bigdata/flume/flume/flumeServer.log**. Most data transmission exceptions and data transmission failures are recorded in logs. You can run the following command:

tailf /var/log/Bigdata/flume/flume/flumeServer.log

- Problem: After the configuration file is uploaded, an exception occurs. After the configuration file is uploaded again, the scenario requirements are still not met, but no exception is recorded in the log.

  Solution: Restart the Flume process, run the **kill -9** *Process code* to kill the process code, and view the logs.

- Issue: **"java.lang.IllegalArgumentException: Keytab is not a readable file: /opt/test/conf/user.keytab"** is displayed when HDFS is connected.

  Solution: Grant the read and write permissions to the Flume running user.

- Problem: The following error is reported when the Flume client is connected to Kafka:

  ```
  Caused by: java.io.IOException: /opt/FlumeClient/fusioninsight-flume-1.9.0/cof//jaas.conf (No such file or directory)
  ```

  Solution: Add the **jaas.conf** configuration file and save it to the **conf** directory of the Flume client.

vi jaas.conf

```
KafkaClient {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
keyTab="/opt/test/conf/user.keytab"
principal="flume_hdfs@<System domain name>"
useTicketCache=false
storeKey=true
debug=true;
};
```

Values of **keyTab** and **principal** vary depending on the actual situation.

- Problem: The following error is reported when the Flume client is connected to HBase:
  ```
  Caused by: java.io.IOException: /opt/FlumeClient/fusioninsight-flume-1.9.0/cof//jaas.conf (No such file or directory)
  ```
  Solution: Add the **jaas.conf** configuration file and save it to the **conf** directory of the Flume client.

  vi jaas.conf

  ```
  Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  keyTab="/opt/test/conf/user.keytab"
  principal="flume_hbase@<System domain name>"
  useTicketCache=false
  storeKey=true
  debug=true;
  };
  ```

  Values of **keyTab** and **principal** vary depending on the actual situation.
- Question: After the configuration file is submitted, the Flume Agent occupies resources. How do I restore the Flume Agent to the state when the configuration file is not uploaded?

  Solution: Submit an empty **properties.properties** file.

7.9.2 How Do I Use Environment Variables in the Flume Configuration File

Step 1 Log in to the node where the Flume client is installed as user **root**.

Step 2 Switch to the following directory:

cd *Flume client installation directory*/**fusioninsight-flume**-*Flume component version*/**conf**

Step 3 Add environment variables to the **flume-env.sh** file in the directory.

- Format:
  ```
  export Variable name=Variable value
  ```
- Example:
  ```
  JAVA_OPTS="-Xms2G -Xmx4G -XX:CMSFullGCsBeforeCompaction=1 -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:+UseCMSCompactAtFullCollection -DpropertiesImplementation=org.apache.flume.node.EnvVarResolverProperties"
  export TAILDIR_PATH=/tmp/flumetest/201907/20190703/1/.*log.*
  ```

Step 4 Restart the Flume instance process.

1. Log in to FusionInsight Manager.

2. Choose **Cluster** > **Services** > **Flume**. On the page that is displayed, click the **Instance** tab, select all Flume instances, and choose **More** > **Restart Instance**. In the displayed **Verify Identity** dialog box, enter the password, and click **OK**.

NOTICE

Do not restart the Flume service on FusionInsight Manager after **flume-env.sh** takes effect on the server. Otherwise, the user-defined environment variables will lost. You only need to restart the corresponding instances on FusionInsight Manager.

Step 5 In the *Flume client installation directory*/**fusioninsight-flume-***Flume component version number*/**conf/properties.properties** configuration file, reference variables in the **${***Variable name***}** format. The following is an example:

```
client.sources.s1.type = TAILDIR
client.sources.s1.filegroups = f1
client.sources.s1.filegroups.f1 = ${TAILDIR_PATH}
client.sources.s1.positionFile = /tmp/flumetest/201907/20190703/1/taildir_position.json
client.sources.s1.channels = c1
```

NOTICE

- Ensure that **flume-env.sh** takes effect before you go to **Step 5** to configure the **properties.properties** file.
- If you configure file on the local host, upload the file on FusionInsight Manager by performing the following steps. The user-defined environment variables may be lost if the operations are not performed in the correct sequence.
  1. Log in to FusionInsight Manager.
  2. Choose **Cluster** > **Services** > **Flume**. On the page that is displayed, click the **Configurations** tab, select the Flume instance, and click **Upload File** next to **flume.config.file** to upload the **properties.properties** file.

----End

7.9.3 How Do I Develop a Third-Party Flume Plug-in

Step 1 Install the Flume client, for example, in the **/opt/flumeclient** directory.

Step 2 Compress the self-developed code into a JAR package.

Step 3 Create a directory for the plug-in.

1. Go to *Flume client installation directory*/**fusionInsight-flume-*/plugins.d** and run the following command to create a directory. The directory name can be changed based on the site requirements.

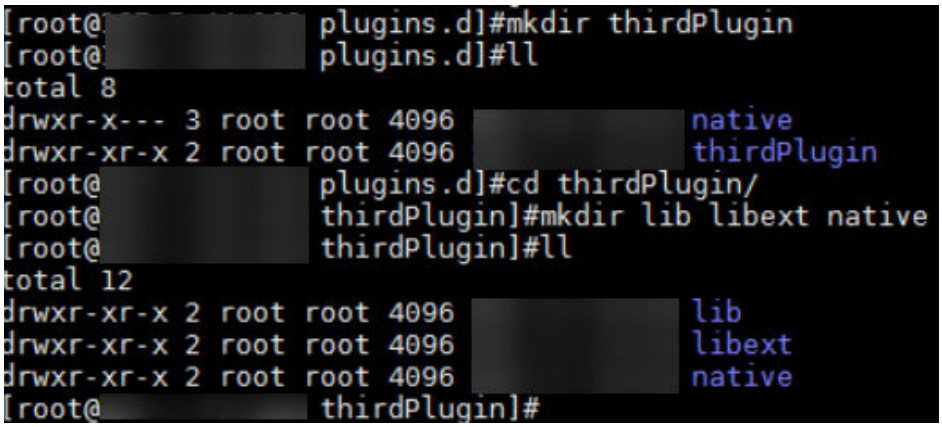
   cd /opt/flumeclient/fusioninsight-flume-1.9.0/plugins.d

   mkdir thirdPlugin

   cd thirdPlugin

   mkdir lib libext native

   The command output is displayed as follows:



2. Place the third-party JAR package in *Flume client installation directory*/**fusionInsight-flume-*/plugins.d/thirdPlugin/lib**. If the JAR package depends on other JAR packages, place the dependent JAR packages in *Flume client installation directory*/**fusionInsight-flume-*/plugins.d/thirdPlugin/libext**. Stores local library files in *Flume client installation directory*/**fusionInsight-flume-*/plugins.d/thirdPlugin/native**.

Step 4 Configure the *Flume client installation directory*/**fusionInsight-flume-*/conf/properties.properties** file.

For details about how to set parameters in the **properties.properties** file, see the parameter list in the **properties.properties** file in the corresponding typical scenario **Configuring a Non-Encrypted Flume Data Collection Task** and **Configuring an Encrypted Flume Data Collection Task**.

----End

7.9.4 How Do I Configure a Custom Flume Script

Flume supports customized scripts to be run before or after transmission for making preparations.

The Flume client is not installed

Step 1 Obtain the software package.

Log in to the FusionInsight Manager. Choose **Cluster** > *Name of the target cluster* > **Services** > **Flume**. On the Flume service page that is displayed, choose **More** > **Download Client** in the upper right corner and set **Client Type** to **Complete Client** to download the Flume service client file.

The file name of the client is **FusionInsight_Cluster_**<*Cluster ID*>**_Flume_Client.tar**. This section takes the client file **FusionInsight_Cluster_1_Flume_Client.tar** as an example.

Step 2 Upload the software package. Upload the software package to a directory, for example, **/opt/client**, on the node where the Flume client is to be installed as user **user**.

NOTE

user is the user who installs and runs the Flume client.

Step 3 Decompress the software package.

Log in to the node where the Flume service client is to be installed as user **user**. Go to the directory where the installation package is installed, for example, **/opt/client**, and run the following command to decompress the installation package to the current directory:

cd /opt/client

tar -xvf FusionInsight_Cluster_1_Flume_Client.tar

Step 4 Verify the software package.

Run the **sha256sum -c** command to verify the decompressed file. If **OK** is returned, the verification is successful. Example:

sha256sum -c FusionInsight_Cluster_1_Flume_ClientConfig.tar.sha256

```
FusionInsight_Cluster_1_Flume_ClientConfig.tar: OK
```

Step 5 Decompress the package.

tar -xvf FusionInsight_Cluster_1_Flume_ClientConfig.tar

Step 6 Configure **client.per-check.shell** in the **/opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient/flume/conf/flume-check.properties** file on the client to point to the absolute path of **plugin.sh**.

The configuration is as follows:

client.per-check.shell=/opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient/flume/plugins.s/plugin.sh

plugins = com.huawei.flume.services.FlumePreTransmitService

flume.check.default.interval = 15

Step 7 Configure the **/opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient/flume/conf/plugin.conf** file and define the scripts to be invoked and related parameters.

The configuration is as follows:

RUN_PLUGIN="PLUGIN_LIST_1"

LOG_TO_HDFS_PATH="/yxs"

LOG_TO_HDFS_ENCODE_PATH="${LOG_TO_HDFS_PATH}/Flume_Encoded/"

PLUGIN_LINK_DIR="/tmp/yxs1"

PLUGIN_MV_TARGET_DIR="/tmp/yxs2"

PLUGIN_SUFFIX="COMPLETED"

PLUGIN_LIST_1="mv_complete.sh --linkdir ${PLUGIN_LINK_DIR} --mvtargetdir ${PLUGIN_MV_TARGET_DIR} --suffix ${PLUGIN_SUFFIX}"

Step 8 Install and start the Flume client. For details about how to install the client, see **Installing the Flume Client**.

----End

The Flume client has been installed

Step 1 Configure **client.per-check.shell** in the **flume-check.properties** file on the client to point to the absolute path of **plugin.sh**.

For example, if the Flume client installation path is **/opt/FlumeClient**, the **flume-check.properties** file is stored in the **/opt/FlumeClient/fusioninsight-flume-1.9.0/conf** directory,

The configuration is as follows:

client.per-check.shell=/opt/FlumeClient/fusioninsight-flume-1.9.0/plugins.s/plugin.sh

plugins = com.huawei.flume.services.FlumePreTransmitService

flume.check.default.interval = 15

Step 2 Configure **plugin.conf** to define the script to be invoked and related parameters.

For example, if the Flume client installation path is **/opt/FlumeClient**, the **plugin.conf** file is stored in the **/opt/FlumeClient/fusioninsight-flume-1.9.0/conf** directory,

The configuration is as follows:

RUN_PLUGIN="PLUGIN_LIST_1"

LOG_TO_HDFS_PATH="/yxs"

LOG_TO_HDFS_ENCODE_PATH="${LOG_TO_HDFS_PATH}/Flume_Encoded/"

PLUGIN_LINK_DIR="/tmp/yxs1"

PLUGIN_MV_TARGET_DIR="/tmp/yxs2"

PLUGIN_SUFFIX="COMPLETED"

PLUGIN_LIST_1="mv_complete.sh --linkdir ${PLUGIN_LINK_DIR} --mvtargetdir ${PLUGIN_MV_TARGET_DIR} --suffix ${PLUGIN_SUFFIX}"

Step 3 Run the following command in the **bin** directory, for example, **/opt/FlumeClient/fusioninsight-flume-1.9.0/bin**, of the client installation path to restart the Flume client:

./flume-manage.sh restart

----End

8 Using Guardian

8.1 Guardian Log Overview

Log Description

Log path: **/var/log/Bigdata/guardian/token-server**

Log archive rule: The automatic compression and archive function is enabled for Guardian run logs. When the total size of all log files exceeds 50 MB (configurable), the log files are automatically compressed into a package named in the format of **token-server.log.[***ID***]**. A maximum of 20 latest compressed files are retained. The number of compressed files and compression threshold can be configured.

Table 8-1 Guardian log list

Log Type	Log File Name	Description	Searchable on Manager
Run log	token-server.log	Guardian run log	Yes
	startDetail.log	Guardian service prestart log	Yes
	stopDetail.log	Guardian service stop log	Yes
	gc.log	Guardian service GC log	No

Log Levels

Table **Table 8-2** describes the log levels provided by Guardian.

The log levels are ERROR, WARN, INFO, and DEBUG in descending order of priority. Only logs whose levels are higher than or equal to the specified level are recorded. The higher the log level specified, the fewer the logs are recorded.

Table 8-2 Log levels

Level	Description
ERROR	Logs of this level record error information about system running
WARN	Logs of this level record exception information about the current event processing
INFO	Logs of this level record normal running status information about the system and events
DEBUG	Logs of this level record the system information and system debugging information

To modify log levels, perform the following operations:

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **Guardian**. Click **Configurations** then **All Configurations**.

Step 3 On the menu bar on the left, select the log menu of the target role.

Step 4 Select a desired log level.

Step 5 Click **Save** then **OK**.

NOTE

The configurations take effect immediately without the need to restart the service.

----End

Log Format

Table 8-3 describes the Guardian log format.

Table 8-3 Log format

Log Type	Format	Example
Run log	<yyyy-MM-dd HH:mm:ss,SSS>|<Log level>|<Name of the thread that generates the log>|<Location where the log event occurs>|< Location of the log event >|<Message in the log>	2024-05-23 09:54:11,696 | INFO | pool-12-thread-1-EventThread:53211544 | mrs.guardian.shaded.org.apache.curator.framework.state.ConnectionStateManager: | ConnectionStateManager.java:252 | State change: CONNECTED

9 Using HBase

9.1 Creating an HBase Permission Role

Scenario

Create and configure an HBase role on Manager as an MRS cluster administrator. The HBase role can set HBase administrator permissions and read (R), write (W), create (C), execute (X), or manage (A) permissions for HBase tables and column families.

Users can create a table, query/delete/insert/update data, and authorize others to access HBase tables after they set the corresponding permissions for the specified databases or tables on HDFS.

NOTE

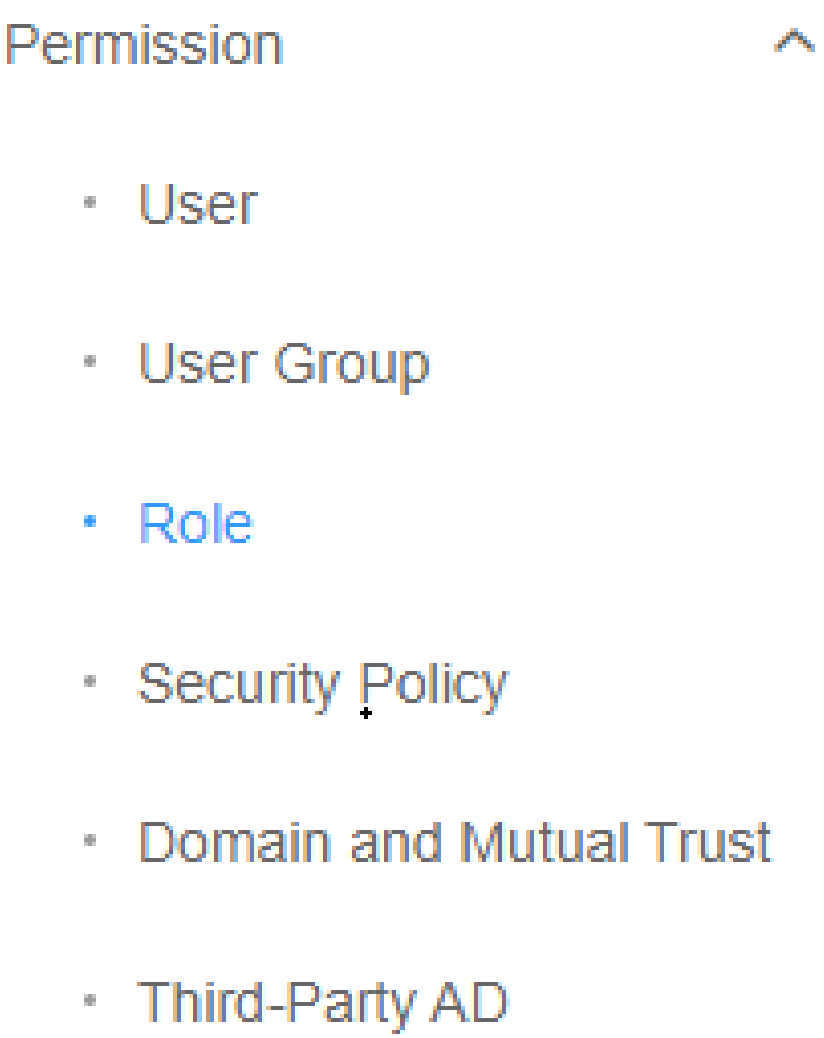
- Only clusters with Kerberos authentication enabled (security mode) support the creation of HBase roles.
- If the current component uses Ranger for permission control, you need to configure related policies based on Ranger for permission management. For details, see **Adding a Ranger Access Permission Policy for HBase**.

Prerequisites

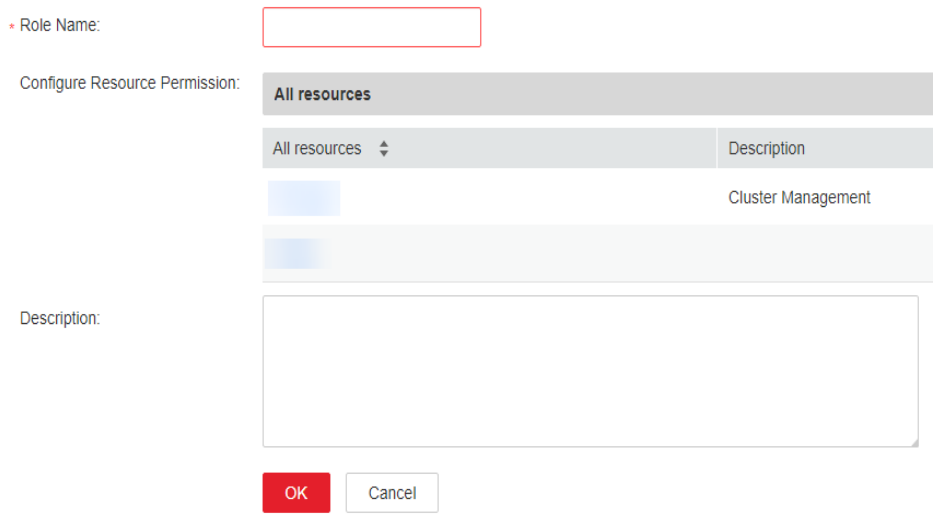
- The MRS cluster administrator has understood service requirements.
- You have logged in to Manager.

Creating an HBase Role

Step 1 On Manager, choose **System** > **Permission** > **Role**.



Step 2 On the displayed page, click **Create Role** and enter a **Role Name** and **Description**.



Step 3 Set **Permission**. For details, see **Table 9-1**.

HBase permissions:

- HBase Scope: Authorizes HBase tables. The minimum permission is read (R) and write (W) for columns.
- SUPER_USER_GROUP: HBase administrator permissions.

NOTE

Users have the read (R), write (W), create (C), execute (X), and administrate (A) permissions for the tables created by themselves.

Table 9-1 Setting HBase role resource permissions

Task	Role Authorization
Setting the HBase administrator permission	In **Configure Resource Permission**, choose *Name of the desired cluster* > **HBase** and select **HBase Administrator Permission**.
Setting the permission for users to create tables	1. In **Configure Resource Permission**, choose *Name of the desired cluster* > **HBase** > **HBase Scope**. 2. Click **global**. 3. In the **Permission** column of the specified namespace, select **Create** and **Execute**. For example, select **Create** and **Execute** for the default namespace **default**.
Setting the permission for users to write data to tables	1. In **Configure Resource Permission**, choose *Name of the desired cluster* > **HBase** > **HBase Scope** > **global**. 2. In the **Permission** column of the specified namespace, select **Write**. For example, select **Write** for the default namespace **default**. By default, HBase sub-objects inherit the permission from the parent object.
Setting the permission for users to read data from tables	1. In **Configure Resource Permission**, choose *Name of the desired cluster* > **HBase** > **HBase Scope** > **global**. 2. In the **Permission** column of the specified namespace, select **Read**. For example, select **Read** for the default namespace **default**. By default, HBase sub-objects inherit the permission from the parent object.

Task	Role Authorization
Setting the permission for users to manage namespaces or tables	1. In **Configure Resource Permission**, choose *Name of the desired cluster* > **HBase** > **HBase Scope** > **global**. 2. In the **Permission** column of the specified namespace, select **admin**. For example, select **admin** for the default namespace **default**.
Setting the permission for reading data from or writing data to columns	1. In **Configure Resource Permission**, select *Name of the desired cluster* > **HBase** > **HBase Scope** > **global** and click the specified namespace to display the tables in the namespace. 2. Click a table. 3. Click a column family. 4. Confirm whether you want to create a role? – If yes, enter the column name in the **Resource Name** text box. Use commas (,) to separate multiple columns. Select **Read** or **Write**. If there are no columns with the same name in the HBase table, a newly created column with the same name as the existing column has the same permission as the existing one. The column permission is set successfully. – If no, modify the column permission of the existing HBase role. The columns for which the permission has been separately set are displayed in the table. Go to **Step 3.5**. 5. To add column permissions for a role, enter the column name in the **Resource Name** text box and set the column permissions. To modify column permissions for a role, enter the column name in the **Resource Name** text box and set the column permissions. Alternatively, you can directly modify the column permissions in the table. If the column permissions are modified in the table and column permissions with the same name are added, the settings cannot be saved. You are advised to modify the column permission of a role directly in the table. The search function is supported.

Step 4 Click **OK**, and return to the **Role** page.

----End

9.2 Using the HBase Client

Scenario

This section describes how to use the HBase client in an O&M scenario or a service scenario.

Video Tutorial

This tutorial demonstrates how to create a table, insert data into it, and modify the table data by logging in to the HBase client after setting up an MRS cluster.

NOTE

The UI may vary depending on the version. This tutorial is for reference only.

Prerequisites

- The client has been installed. For example, the installation directory is **/opt/hadoopclient**. The client directory in the following operations is only an example. Change it to the actual installation directory.
- Service component users have been created by the MRS cluster administrator.

  A machine-machine user needs to download the **keytab** file and a human-machine user needs to change the password upon the first login.
- If a non-**root** user uses the HBase client, ensure that the owner of the HBase client directory is this user. Otherwise, run the following command to change the owner.

  chown user:group -R *Client installation directory*/**HBase**

Using the HBase Client

Step 1 Install the client. For details, see **Installing a Client**.

Step 2 Log in to the node where the client is installed as the client installation user.

Step 3 Run the following command to go to the client directory:

cd /opt/hadoopclient

Step 4 Run the following command to configure environment variables:

source bigdata_env

Step 5 If Kerberos authentication has been enabled for the current cluster, run the following command to authenticate the current user. The current user must have the permission to create HBase tables. For details about how to configure a role with the corresponding permissions, see**Managing Roles**. To bind a role to a user, see **Creating a User**. If Kerberos authentication is disabled for the current cluster, skip this step.

kinit *Component service user*

For example, **kinit hbaseuser**.

Step 6 Run the following HBase client command:

hbase shell

----End

Common HBase client commands

The following table lists common HBase client commands. For more commands, see **http://hbase.apache.org/2.2/book.html**.

Table 9-2 HBase client commands

Command	Description
create	Used to create a table, for example, **create 'test', 'f1', 'f2', 'f3'**.
disable	Used to disable a specified table, for example, **disable 'test'**.
enable	Used to enable a specified table, for example, **enable 'test'**.
alter	Used to alter the table structure. You can run the **alter** command to add, modify, or delete column family information and table-related parameter values, for example, **alter 'test', {NAME => 'f3', METHOD => 'delete'}**.
describe	Used to obtain the table description, for example, **describe 'test'**.
drop	Used to delete a specified table, for example, **drop 'test'**. Before deleting a table, you must stop it.
put	Used to write the value of a specified cell, for example, **put 'test','r1','f1:c1','myvalue1'**. The cell location is unique and determined by the table, row, and column.
get	Used to get the value of a row or the value of a specified cell in a row, for example, **get 'test','r1'**.
scan	Used to query table data, for example, **scan 'test'**. The table name and scanner must be specified in the command.

9.3 Using HBase for Offline Data Analysis

HBase is a column-based distributed storage system that features high reliability, performance, and scalability. This section describes how to use HBase from scratch, including create a table using the client, insert data in the table, modify the table, read data from the table, delete table data, and delete the table.

Background

Suppose a user develops an application to manage users who use service A in an enterprise. The procedure of operating service A on the HBase client is as follows:

- Create the **user_info** table.
- Add users' educational backgrounds and titles to the table.
- Query user names and addresses by user ID.
- Query information by user name.
- Deregister users and delete user data from the user information table.
- Delete the user information table after service A ends.

Table 9-3 User information

ID	Name	Gender	Age	Address
12005000201	A	Male	19	City A
12005000202	B	Female	23	City B
12005000203	C	Male	26	City C
12005000204	D	Male	18	City D
12005000205	E	Female	21	City E
12005000206	F	Male	32	City F
12005000207	G	Female	29	City G
12005000208	H	Female	30	City H
12005000209	I	Male	26	City I
12005000210	J	Male	25	City J

Prerequisites

The client has been installed in a directory, for example, **/opt/client**. The client directory in the following operations is only an example. Change it to the actual installation directory. Before using the client, download and update the client configuration file, and ensure that the active management node of Manager is available.

Procedure

Step 1 Use the client on the active management node.

1. Install the client. For details, see **Installing a Client**.
2. Log in to the node where the client is installed as the client installation user and run the following command to switch to the client directory:

   cd /opt/client
3. Run the following command to configure environment variables:

   source bigdata_env
4. If Kerberos authentication has been enabled for the current cluster, run the following command to authenticate the current user. The current user must have the permission to create HBase tables. For details about how to

configure a role with the corresponding permissions, see **Creating an HBase Permission Role**. To bind a role to a user, see **Creating a User**. If Kerberos authentication is disabled for the current cluster, skip this step.

kinit *MRS cluster user*

For example, **kinit hbaseuser**.

5. Run the following HBase client command:

   hbase shell

Step 2 Run the following commands on the HBase client to implement service A.

1. Create the **user_info** user information table according to **Table 9-3** and add data to it.

   create '*user_info*',{**NAME** => 'i'}

   For example, to add information about the user whose ID is **12005000201**, run the following commands:

   put '*user_info*','*12005000201*','***i:name***','*A*'

   put '*user_info*','*12005000201*','***i:gender***','*Male*'

   put '*user_info*','*12005000201*','***i:age***','*19*'

   put '*user_info*','*12005000201*','***i:address***','*City A*'

2. Add users' educational backgrounds and titles to the **user_info** table.

   For example, to add educational background and title information about user 12005000201, run the following commands:

   put '*user_info*','*12005000201*','**i:degree**','*master*'

   put '*user_info*','*12005000201*','**i:pose**','*manager*'

3. Query user names and addresses by user ID.

   For example, to query the name and address of user 12005000201, run the following command:

   scan'*user_info*',
   {**STARTROW**=>'*12005000201*',**STOPROW**=>'*12005000201*',**COLUMNS**=>['**i:name**','**i:address**']}

4. Query information by user name.

   For example, to query information about user A, run the following command:

   scan '*user_info*',{**FILTER**=>"SingleColumnValueFilter('i','name',=,'binary:*A*')"}

5. Delete user data from the user information table.

   All user data needs to be deleted. For example, to delete data of user 12005000201, run the following command:

   - Delete all data fields of the user whose ID is 12005000201 in sequence. The following uses the **age** field as an example:

     delete '*user_info*','*12005000201*','i:age'

   - Remove all the data of the user whose ID is 12005000201.

     deleteall '*user_info*','*12005000201*'

6. Delete the user information table.

   disable '*user_info*'

   drop '*user_info*'

----End

9.4 Migrating Data to HBase Using BulkLoad

HBase data is stored in HDFS. To import data, load it from HDFS into an HBase table. Apache HBase provides the Import and ImportTsv tools for batch importing data.

- Import: imports HBase data that has been exported to HDFS using the **org.apache.hadoop.hbase.mapreduce.Import** method.
- ImportTsv: loads data in TSV format to HBase through **org.apache.hadoop.hbase.mapreduce.ImportTsv**.

For details, see **http://hbase.apache.org/2.2/book.html#tools**.

9.5 HBase Data Operations

9.5.1 Creating HBase Indexes for Data Query

Scenario

HBase is a distributed storage database of the key-value type. HIndex enables HBase indexing based on specific column values, making the retrieval of data highly efficient and fast.

Constraints

- Column families are separated by semicolons (;).
- Columns and data types must be contained in square brackets ([]).
- The column data type is specified by using -> after the column name.
- If the column data type is not specified, the default data type (string) is used.
- The number sign (#) is used to separate two index details.
- The following is an optional parameter:

  -Dscan.caching: number of cached rows when the data table is scanned.

  The default value is set to 1000.
- Indexes are created for a single region to repair damaged indexes. This function is not used to generate new indexes.

Create an HBase HIndex

Step 1 Install the HBase client. For details, see **Using the HBase Client**.

Step 2 Log in to the node where the client is installed as the client installation user.

Step 3 Go to the client installation directory, for example, **/opt/client**.

cd /opt/client

Step 4 Run the following command to configure environment variables:

source bigdata_env

Step 5 If the cluster is in security mode, run the following command to authenticate the user. In normal mode, user authentication is not required.

kinit *Component service user*

Step 6 Run the following command to access HIndex:

hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer

Table 9-4 Common HIndex commands

Function	Command
Add Index	TableIndexer-Dtablename.to.index=table1-Dindexspecs.to.add='IDX1=>cf1:[q1->datatype],[q2],[q3];cf2:[q1->datatype],[q2->datatype]#IDX2=>cf1:[q5]'
Create Index	TableIndexer -Dtablename.to.index=table1 -Dindexnames.to.build='IDX1#IDX2'
Delete Index	TableIndexer -Dtablename.to.index=table1 -Dindexnames.to.drop='IDX1#IDX2'
Disable Index	TableIndexer -Dtablename.to.index=table1 -Dindexnames.to.disable='IDX1#IDX2'
Add and Create Index	TableIndexer -Dtablename.to.index=table1 -Dindexspecs.to.add='IDX1=>cf1:[q1->datatype],[q2],[q3];cf2:[q1->datatype],[q2->datatype]#IDX2=>cf1:[q5]' -Dindexnames.to.build='IDX1'
Create Index for a Single Region	TableIndexer -Dtablename.to.index=table1 -Dregion.to.index=regionEncodedName -Dindexnames.to.build='IDX1#IDX2'

NOTE

- **IDX1**: indicates the index name.
- **cf1**: indicates the column family name.
- **q1**: indicates the column name.
- **datatype**: indicates the data type, including String, Integer, Double, Float, Long, Short, Byte and Char.

----End

9.5.2 Configuring the HBase Data Compression and Encoding Formats

Scenario

HBase encodes data blocks in HFiles to reduce duplicate keys in KeyValues, reducing used space. Currently, the following data block encoding modes are supported: NONE, PREFIX, DIFF, FAST_DIFF, and ROW_INDEX_V1. NONE indicates

that data blocks are not encoded. HBase also supports compression algorithms for HFile compression. The following algorithms are supported by default: NONE, GZ, SNAPPY, and ZSTD. NONE indicates that HFiles are not compressed.

The two methods are used on the HBase column family. They can be used together or separately.

Prerequisites

- The HBase client has been installed in a directory, for example, **/opt/client**.
- If Kerberos authentication has been enabled for HBase, you must have the corresponding operation permissions. For example, you must have the creation (C) or administration (A) permission on the corresponding namespace or higher-level items to create a table, and the creation (C) or administration (A) permission on the created table or higher-level items to modify a table. For details about how to grant permissions, see **Creating an HBase Permission Role**.

Configuring the HBase Data Compression and Encoding Formats

Setting data block encoding and compression algorithms during creation

- **Method 1: Using hbase shell**
  a. Log in to the node where the client is installed as the client installation user.
  b. Run the following command to go to the client directory:
     cd /opt/client
  c. Run the following command to configure environment variables:
     source bigdata_env
  d. If the Kerberos authentication is enabled for the current cluster, run the following command to authenticate the user. If Kerberos authentication is disabled for the current cluster, skip this step:
     kinit *Component service user*
     For example, **kinit hbaseuser**.
  e. Run the following HBase client command:
     hbase shell
  f. Create a table.
     create '*t1***', {NAME => '***f1***', COMPRESSION => '***SNAPPY***', DATA_BLOCK_ENCODING => '***FAST_DIFF***'}**

     NOTE

     - *t1*: indicates the table name.
     - *f1*: indicates the column family name.
     - *SNAPPY*: indicates the column family uses the SNAPPY compression algorithm.
     - *FAST_DIFF*: indicates FAST_DIFF is used for encoding.
     - The parameter in the braces specifies the column family. You can specify multiple column families using multiple braces and separate them by commas (,). For details about table creation statements, run the **help 'create'** statement in the HBase shell.

- **Method 2: Using Java APIs**

  The following code snippet shows only how to set the encoding and compression modes of a column family when creating a table. For complete code for creating a table and how to use the code to create a table, see **Creating an HBase Table** in the HBase Development Guide.

```
TableDescriptorBuilder htd = TableDescriptorBuilder.newBuilder(TableName.valueOf("t1"));// Create a descriptor for table t1.
ColumnFamilyDescriptorBuilder hcd = ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes("f1"));// Create a builder for column family f1.
hcd.setDataBlockEncoding(DataBlockEncoding.FAST_DIFF);// Set the encoding mode of column family f1 to FAST_DIFF.
hcd.setCompressionType(Compression.Algorithm.SNAPPY);// Set the compression algorithm of column family f1 to SNAPPY.
htd.setColumnFamily(hcd.build())// Add the column family f1 to the descriptor of table t1.
```

Setting or modifying the data block encoding mode and compression algorithm for an existing table

- **Method 1: Using hbase shell**

  a. Log in to the node where the client is installed as the client installation user.

  b. Run the following command to go to the client directory:

     cd /opt/client

  c. Run the following command to configure environment variables:

     source bigdata_env

  d. If the Kerberos authentication is enabled for the current cluster, run the following command to authenticate the user. If Kerberos authentication is disabled for the current cluster, skip this step:

     kinit *Component service user*

     For example, **kinit hbaseuser**.

  e. Run the following HBase client command:

     hbase shell

  f. Run the following command to modify the table:

     alter '*t1***', {NAME => '***f1***', COMPRESSION => '***SNAPPY***', DATA_BLOCK_ENCODING => '***FAST_DIFF***'}**

- **Method 2: Using Java APIs**

  The following code snippet shows only how to modify the encoding and compression modes of a column family in an existing table. For complete code for modifying a table and how to use the code to modify a table, see **Modifying a Table** in HBase Development Guide.

```
TableDescriptor htd = admin.getDescriptor(TableName.valueOf("t1"));// Obtain the descriptor of table t1.
ColumnFamilyDescriptor originCF = htd.getColumnFamily(Bytes.toBytes("f1"));// Obtain the descriptor of column family f1.
builder.ColumnFamilyDescriptorBuilder hcd = ColumnFamilyDescriptorBuilder.newBuilder(originCF);// Create a builder based on the existing column family attributes.
hcd.setDataBlockEncoding(DataBlockEncoding.FAST_DIFF);// Change the encoding mode of the column family to FAST_DIFF.
hcd.setCompressionType(Compression.Algorithm.SNAPPY);// Change the compression algorithm of the column family to SNAPPY.
admin.modifyColumnFamily(TableName.valueOf("t1"), hcd.build());// Submit to the server to modify the attributes of column family f1.
```


After the modification, the encoding and compression modes of the existing HFile will take effect after the next compaction.

9.6 Enterprise-Class Enhancements of HBase

9.6.1 Configuring HBase Global Secondary Indexes for Faster Queries

9.6.1.1 Introduction to HBase Global Secondary Indexes

Scenarios

HBase secondary indexes can be used to accelerate conditional queries with filters. HBase secondary indexes support local secondary indexes (HIndexes) and global secondary indexes (GSIs). Compared with local indexes (HIndex), global secondary indexes have better query performance and are suitable for scenarios that require high read latency.

HBase global secondary indexes use independent index tables to store index data. When a given query condition can hit an index, a full table query on a data table can be converted into an exact range query on an index table, thereby improving a query speed. After the global secondary index feature is enabled, the code on the application side does not need to be modified, which is easy to use.

NOTE

For MRS 3.3.0 or later, HBase global secondary index is enable by default. To modify the related parameters, log in to FusionInsight Manager, choose **Cluster** > **Services** > **HBase** > **Confiurations** > **All Configurations**, select **RegionServer(Role)** > **Secondary Indexes** and **HMaster(Role)** > **Secondary Indexes**.

HBase global secondary indexes support the following key features:

- Composite Index

  Multiple columns can be specified as index columns (cross-column-family is supported).

- Overwrite Index

  Multiple columns or column families can be specified as overwrite columns or column families and stored in the index table in redundancy mode. This function is used to quickly query non-index columns in index query.

- Index TTL

  Index table TTL is used when TTL is enabled for a data table. To ensure consistency with the data table, the index table TTL automatically inherits the TTL of the index column and overwrites column of the data table and cannot be manually specified.

- Online index change

  Indexes can be created, deleted, and their status can be modified online without affecting data table read and write.

- Online index repair

  When the index data hit by the query is invalid, index repair can be triggered to ensure that the final query result is correct.

- Index Tool

  Supports index consistency check, index repair, index creation, deletion, and status modification, and index data rebuilding.

Restrictions on HBase Global Secondary Indexes

- Application Scenario Restrictions
  - GSI cannot be used together with HIndex. That is, local indexes and global indexes cannot be created in the same data table at the same time.
  - Index tables do not support DR.
  - Rolling upgrade is not supported for index data.
  - DISABLE, DROP, MODIFY, and TRUNCATE cannot be directly performed on index tables.
  - DDL operations on indexes allow you to modify index status, delete indexes, and create indexes. Index definitions cannot be modified. To modify index definitions, delete them and create indexes again.
- Constraints on Index Creation
  - The index name must comply with the regular expression requirements and does not support other characters. The regular expression must support the following characters: [a-zA-Z_0-9-.]:
  - The data table must exist. The index to be created cannot exist.
  - The index table does not support multiple versions.

    Indexes cannot be created on data tables with multiple versions (VERSION>1), and the version of the index table is 1.
  - The number of indexes in a single data table cannot exceed 5.

    You are not advised to create too many indexes for a single data table. If there are too many indexes, the storage cost is high and the write time is long. If more than five indexes need to be created, add the hbase.gsi.max.index.count.per.table parameter to the customized configuration hbase.hmaster.config.expandor of HMaster and set the parameter to a value greater than 5. Restart HMaster for the configuration to take effect.
  - The index name can contain a maximum of 18 characters.

    You are not advised to use an excessively long index name. To create a long index name, add the hbase.gsi.max.index.name.length parameter to the customized configuration hbase.hmaster.config.expandor of HMaster, set the parameter to a value greater than 18, and restart HMaster for the configuration to take effect.
  - Indexes cannot be created for index tables.

    Multiple indexes cannot be created in nested mode. Index tables are used only to accelerate query and do not provide data table functions.
  - Indexes that can be overwritten by existing indexes cannot be created.

    When you create an index, if the existing index can completely overwrite the new index (that is, the created index is a subset of the existing index),

the index cannot be created. Indexes with duplicate functions cause storage waste. For example, index 2 cannot be created if you perform the following operations:

Create a data table: **create't1','cf1'**

Create index 1: **hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='t1' -Dindexspecs.to.add='idx1=>cf1:[q1],[q2]'**

Create index 2: **hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='t1' -Dindexspecs.to.add='idx2=>cf1:[q1]'**

- Indexes with the same name cannot be created in the same data table. Indexes with the same name can be created in different data tables.
- The TTL of the index table column family is inherited from the original table. The TTL of the index column family must be the same.

  The TTLs of all column families in an index table are the same and are inherited from a data table. The TTLs of related column families in the data table must be the same. Otherwise, related indexes cannot be created.
- Other attributes of the user-defined index table are not supported.

- Constraints on Writing Indexes
  - Only the Put/Delete interface can be used to generate index data. When data is written to a data table in other modes (such as Increment, Append, and Bulkload), the corresponding index is not generated.
  - When the index column data is defined as the string type, do not write the special characters \x00 and \x01 (special invisible characters).
  - Do not write data to index columns by specifying timestamps.
- Constraints on Index Query
  - The index status must be ACTIVE during index query.
  - The index query does not support the query by specifying the timestamp range. If you need to query data within the time range by index, add a time column to store the timestamp of the data. Otherwise, the data table will be used for query.
  - Index query does not support range query by specifying StartRow and StopRow. If either of them is specified, the query operation does not use the index query. Instead, the Range Scan+Filter filtering function of the primary table is used.
  - Index query supports only SingleColumnValueFilter. Index acceleration cannot be triggered when other filters are used or no filter condition is used.

9.6.1.2 Creating an HBase Global Secondary Index

Scenarios

- If a large amount of data exists in a table, you can add an index to a column.
- For user tables that do not have indexes, this tool allows you to add and build indexes at the same time.

Creating an HBase Global Secondary Index

Run the following command on the HBase client to add or create an index. After the command is executed, the specified index is added to the table.

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='*table*' **-Dindexspecs.to.add**='*idx1=>cf1:[c1->string],[c2]#idx2=>cf2:[c1->string],[c2]#idx3=>cf1:[c1];cf2:[c1]*' **-Dindexspecs.covered.family.to.add**='*idx2=>cf1*' **-Dindexspecs.covered.to.add**='*idx1=>cf1:[c3],[c4]*' **-Dindexspecs.coveredallcolumn.to.add**='*idx3=>true*' **-Dindexspecs.splitkeys.to.set**='*idx1=>[\x010,\x011,\x012]#idx2=>[\x01a,\x01b,\x01c]#idx3=>[\x01d,\x01e,\x01f]*'

The parameters are described as follows:

- **tablename.to.index**: Name of the data table for which an index is created

  NOTE

  When this parameter is used to create an index, if the data table is empty, the created index will be in **ACTIVE** state. Otherwise, the index will be in **INACTIVE** state.

- **indexspecs.to.addandbuild** (optional): Generated index data during data table creation. If the data table is too large, enabling this parameter is **not recommended**. Use an index data generation tool instead.

  NOTE

  This parameter and **tablename.to.index** cannot be used at the same time. When this parameter is used, the index will be in **BUILDING** state. After the index data is generated, the index will be in **ACTIVE** state.

- **indexspecs.to.add**: Mapping between the index name and the column in the corresponding data table (definition of index column)
- **indexspecs.covered.to.add** (optional): Column of the data table that is redundantly stored in an index (definition of overwrite column)
- **indexspecs.covered.family.to.add** (optional): Column family of the data table that is redundantly stored in an index table (definition of overwrite column)
- **indexspecs.coveredallcolumn.to.add** (optional): All data in a data table that is redundantly stored in an index table (definition of overwrite all columns)
- **indexspecs.splitkeys.to.set** (optional): Pre-partition split point of an index table. **Specify this parameter** in case the Region index table becomes a hotspot. The format of pre-partition is as follows:
  - **'#'**: separate indexes
  - **'[]'**: contain splitkeys
  - **','**: separate splitkeys

    NOTE

    Each splitkey of the pre-partition must start with **\x01**.

The parameters in the preceding command are described as follows:

- **idx1**, **idx2**, and **idx3**: index names
- **cf1** and **cf2**: column family names

- **c1**, **c2**, **c3**, and **c4**: column names
- **string**: data type. The value can be **STRING**, **INTEGER**, **FLOAT**, **LONG**, **DOUBLE**, **SHORT**, **BYTE**, or **CHAR**.

NOTE

- '**#**' is used to separate indexes, '**;**' is used to separate column families, and '**,**' is used to separate column qualifiers.
- The column name and its data type must be included in '**[]**'.
- Column names and their data types are separated by '**->**'.
- If the data type of a specific column is not specified, the default data type (string) is used.

Deleting an HBase Global Secondary Index

Run the following command on the HBase client to delete an index:

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='*table*' **-Dindexnames.to.drop**='*idx1#idx2*'

The parameters are described as follows:

- tablename.to.index: indicates the name of the table where the index to be deleted is located.
- indexnames.to.drop: indicates the name of the index to be deleted. You can specify multiple indexes and separate them with number signs (#).

9.6.1.3 Querying an HBase Global Secondary Index

Scenarios

You can use the global secondary index tool to view the definition and status of indexes related to a data table in batches.

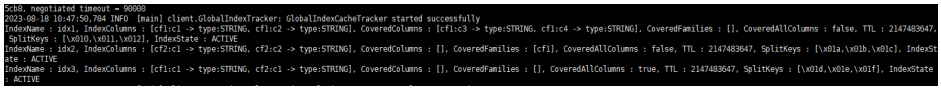
Querying an HBase Global Secondary Index

Run the following command on the HBase client to view the definition and status of the index:

Indicates the name of the hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.show=' data table.'

Figure 9-1 shows the query result. The index column definition, overwrite column definition, TTL, pre-partition information, and index status are printed.

Figure 9-1 Index query result



9.6.1.4 Changing Status of HBase Global Secondary Indexes

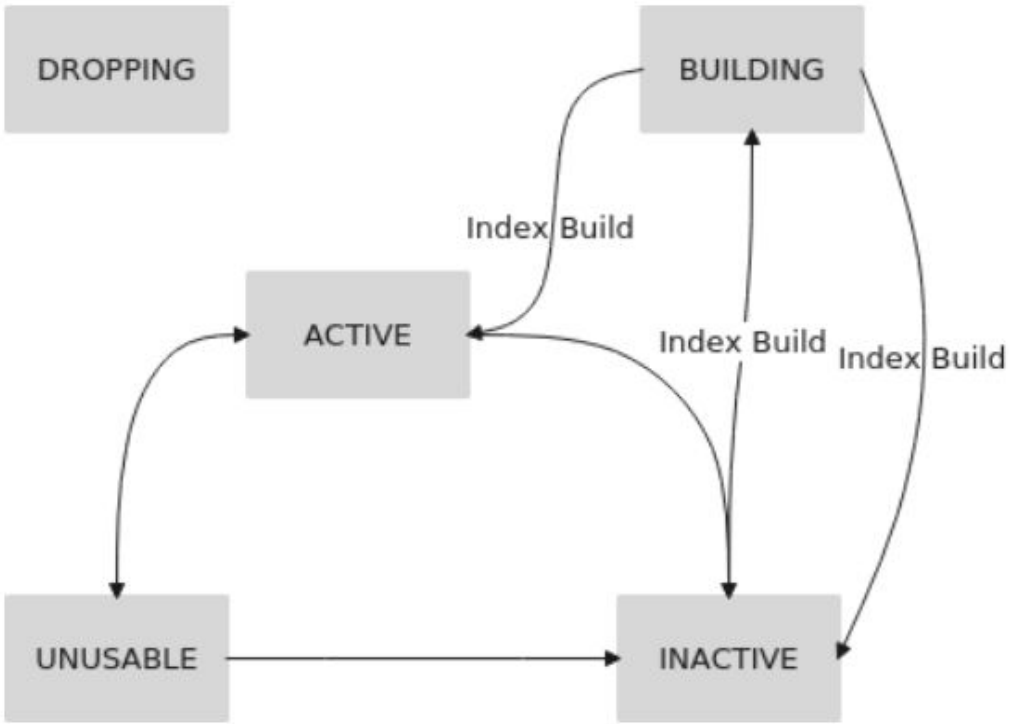
Index Status

The index status reflects the current index usage. The global secondary index supports the following five states:

- ACTIVE: The index is normal and can be read and written normally.
- UNUSABLE: The index is disabled. Index data can be written normally and cannot be used for query.
- INACTIVE: The index is abnormal. The index data is inconsistent with that in the data table. The index data for generating the index is skipped. The index cannot be used during data query.
- BUILDING: Index data is generated in batches. After the index data generation tool is executed, the index data is automatically switched to the ACTIVE state. In this state, data can be read and written properly.
- DROPPING: The index is being deleted. The index data generated for the index is skipped. The index cannot be used during data query.

The index status can be changed based on the tool. **Figure 9-2** describes the status change.

Figure 9-2 Index status transition



Scenarios

You can use the global secondary index tool to disable or enable an index.

Changing Status of HBase Global Secondary Indexes

Run the following command on the HBase client to disable or enable an index:

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='*table*' **-D**[idx_state_opt]='*idx1*'

The related parameters are described as follows:

- tablename.to.index: indicates the name of the data table whose index status needs to be changed.
- idx_state_opt: target status of the index to be modified. The options are as follows:
  - indexnames.to.inactive: changes the status of a specified index to INACTIVE.
  - indexnames.to.active: changes the status of a specified index to ACTIVE.
  - indexnames.to.unusable: converts the specified index to the UNUSABLE state.

For example, to change the status of the idx1 index in the ACTIVE table to UNUSABLE, run the following command:

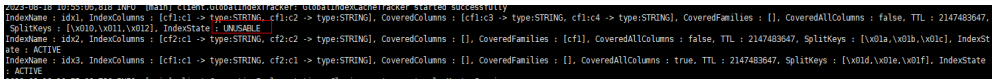
hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='table' -Dindexnames.to.unusable='idx1'

After the command is executed successfully, check the index information again.

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.show='table'

As shown in **Figure 9-3**, the index status of idx1 has been changed.

Figure 9-3 idx1 index status



9.6.1.5 Creating HBase Global Secondary Indexes in Batches

Scenarios

If a large amount of data exists in a user table, index data of the existing data can be constructed in batches based on MapReduce tasks.

Creating HBase Global Secondary Indexes in Batches

NOTICE

- Only indexes in INACTIVE state can be built in batches. To rebuild index data, change the index status first.
- If a data table contains a large amount of data, the construction takes a long time. You are advised to run the nohup command in the background to prevent the operation from being interrupted unexpectedly.

Run the following command on the HBase client to create index data for existing data in batches:

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='*table*' **-Dindexnames.to.build**='*idx1*'

The related parameters are described as follows:

- tablename.to.index: indicates the name of the data table whose index status needs to be changed.
- indexnames.to.build: specifies the names of the indexes for which data needs to be generated in batches. You can specify multiple indexes and separate them with number signs (#).
- hbase.gsi.cleandata.enabled (optional): indicates whether to clear the index table before creating index data. The default value is false.
- (Optional) hbase.gsi.cleandata.timeout: timeout interval for clearing the index table before creating index data. The default value is 1800, in seconds.

9.6.1.6 Checking HBase Global Secondary Index Data Consistency

Scenarios

You can use the global secondary index tool to check the consistency between user data and index data. If the index data is inconsistent with user data, this tool can be used to rebuild index data.

Checking HBase Global Secondary Index Data Consistency

Run the following command on the HBase client to check data consistency. If data is inconsistent, index data will be rebuilt. The consistency check result is saved to the {NameSpace where the data table is located}:GSI_INCONSISTENCY_TABLE table.

hbase org.apache.hadoop.hbase.hindex.global.tools.GlobalHIndexConsistencyTool -dt *table1* **-n** *idx3* **-src** *BOTH* **-r**

The parameters are described as follows:

- -dt,--data-table: indicates the name of the data table to be checked.
- -n,--index-name: indicates the name of the index for which the consistency check is to be performed.
- -src,--source: indicates the check mode. The default value is BOTH. The following modes are supported:
  - INDEX_TABLE_SOURCE: The index table is used as the source table.
  - DATA_TABLE_SOURCE: The data table is used as the source table.
  - BOTH: Both index tables and data tables are source tables.
- -r,--repair: index data repair option. If this parameter is added, the index data is repaired after the check.
- (Optional) -sc,--scan-caching: size of scan caching in a MapReduce job for consistency check or repair.

9.6.1.7 Querying HBase Table Data with Global Secondary Indexes

Index-based Query HBase Table Data

In a user table with an index, you can use SingleColumnValueFilter to query data. When the query condition can match the index, the query speed is much faster than that of the original table query.

The index matching rules are as follows:

- Query by Multiple AND Conditions
  - When the columns used for query contain at least the first column of the index, using the index improves the query performance.

    For example, create a combination index for C1, C2, and C3.

    The index takes effect in the following situations:

    Filter_Condition (IndexCol1) AND Filter_Condition (IndexCol2) AND Filter_Condition (IndexCol3)

    Filter_Condition (IndexCol1) AND Filter_Condition (IndexCol2)

    Filter_Condition (IndexCol1) AND Filter_Condition (IndexCol3)

    Filter_Condition (IndexCol1)

    The index does not take effect in the following situations:

    Filter_Condition (IndexCol2) AND Filter_Condition (IndexCol3)

    Filter_Condition (IndexCol2)

    Filter_Condition (IndexCol3)
  - When you use **Index Column AND Non-Index Column** for filtering in the query, the index can improve query performance. If a non-index column hits an overwrite column, the query performance is optimal. If a non-index column needs to be frequently queried, you are advised to define it as an overwrite column. The following is an example:

    Filter_Condition (IndexCol1) AND Filter_Condition (NonIndexCol1)

    Filter_Condition (IndexCol1) AND Filter_Condition (IndexCol2) AND Filter_Condition (NonIndexCol1)
  - When multiple columns are used for query, you can specify a value range for only the last column in the combination index and set other columns to specified values

    For example, create a combination index for C1, C2, and C3. In a range query, only the value range of C3 can be set. The filter criteria are "C1 = *XXX*, C2 = *XXX*, and C3 = *Value range*".
- Query by Multiple OR Conditions

  For example, create a combination index for C1, C2, and C3.
  - When only the first field in the index column is filtered (range filtering is supported), the index can be used to improve the query performance.

    Filter_Condition (IndexCol1) OR Filter_Condition (IndexCol1) OR Filter_Condition (IndexCol1)
  - When non-index and non-index columns are filtered, indexes cannot be hit, and the query performance is not improved.

    Filter_Condition (IndexCol1) OR Filter_Condition (NonIndexCol1)

– During combined query, if the outermost layer contains the OR condition, the index cannot be matched, and the query performance is not improved.

  Filter_Condition (IndexCol1) OR Filter_Condition (NonIndexCol1)

  (Filter_Condition(IndexCol1)AND Filter_Condition (IndexCol2))OR(Filter_Condition(NonIndexCol1))

NOTE

Reduce the use of OR conditions, especially OR conditions and range conditions. When indexes are matched, large-scale query is performed and the query speed is slow.

9.6.2 Configuring HBase Local Secondary Indexes for Faster Queries

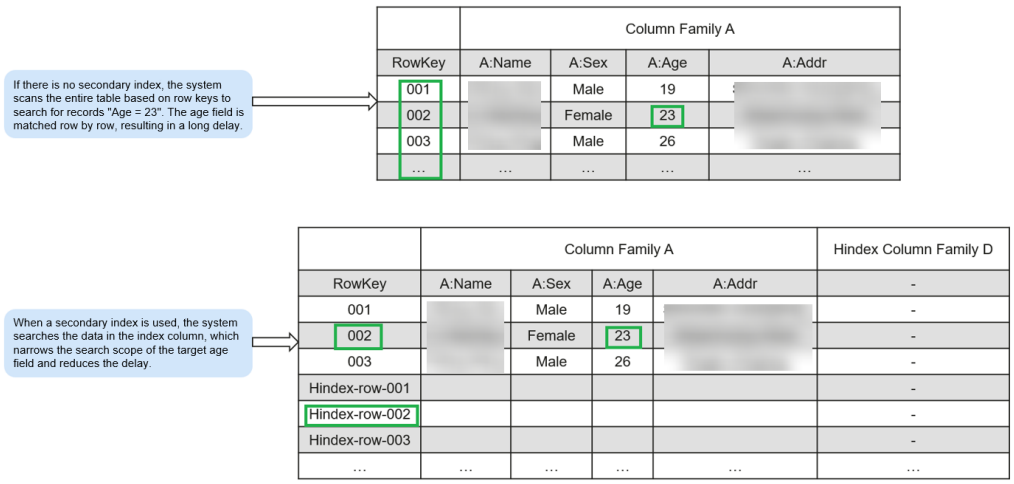
9.6.2.1 Introduction to HBase Local Secondary Indexes

Scenarios

HBase is a distributed storage database of the Key-Value type. Data in tables is sorted by dictionary based on row keys. If you query data by specifying a row key or scan data in a specific row key range, HBase can help you quickly locate the data to be read. In most cases, you need to query data whose column value is *XXX*. HBase provides the filter function to enable you to query data with a specific column value. All data is scanned in the sequence of row keys and is matched with the specific column value until the required data is found. To obtain the required data, the filter will scan some unnecessary data. As a result, the filter function cannot meet the requirements for high-performance, frequent queries.

HBase HIndex is designed to address these issues. HBase HIndex provides HBase with the capability of indexing based on specific column values, making queries faster.

Figure 9-4 HBase HIndex

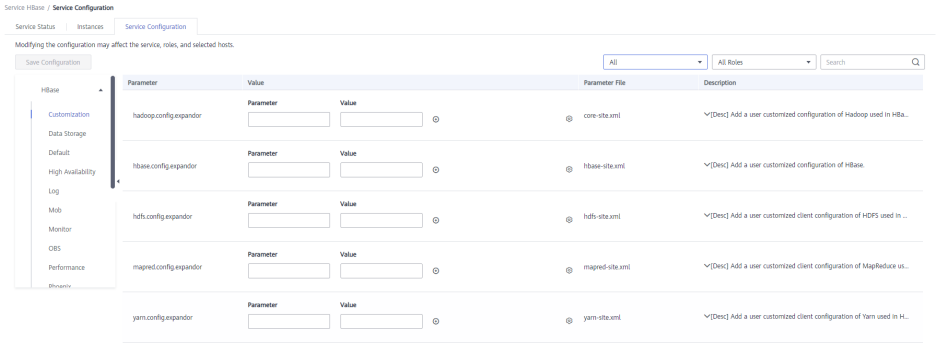


NOTE

- Rolling upgrade is not supported for index data.
- Composite index: You must add or delete all columns that participate in composite indexes. Otherwise, the data may be inconsistent.
- You should not explicitly configure any split policy to a data table where an index has been created.
- The mutation operations are not supported, such as increment and append.
- Index of the column with **maxVersions** greater than 1 is not supported.
- The value size of a column for which an index is added cannot exceed 32 KB.
- When the user data is deleted because TTL of the column family is invalid, the corresponding index data will not be deleted immediately. The index data will be deleted during major compaction.
- After an index is created, the TTL of the user column family must not be changed.
  - If the TTL of the column family is changed to a larger value after an index is created, delete the index and create one again. Otherwise, some generated index data may be deleted before the deletion of user data.
  - If the TTL of the column family is changed to a smaller value after an index is created, the index may be deleted after the deletion of user data.
- After disaster recovery is enabled for HBase tables, a secondary index is created in the active cluster and index table changes are not automatically synchronized to the standby cluster. To implement disaster recovery in this case, perform the following operations:
  1. After the secondary index is created in the active table, create a secondary index with the same schema and name using the same method in the standby cluster.
  2. In the active cluster, manually set **REPLICATION_SCOPE** of the index column family (default value: **d**) to **1**.

Configuring the HBase Local Secondary Index

1. Log in to the MRS console, click the cluster name in the existing cluster list, and choose **Components**.
2. In the component list, choose **HBase** > **Service Configuration**. In the drop-down list, switch **Basic Configuration** to **All Configurations**. The **All Configurations** page is displayed.



3. View parameters on the HBase configurations page.

Table 9-5 Parameters related to HBase Index

Navigation Path	Parameter	Default Value	Description
HMaster > System	hbase.coprocessor.master.classes	org.apache.hadoop.hbase.hindex.server.master.HIndexMasterCoprocessor,com.xxx.hadoop.hbase.backup.services.RecoveryCoprocessor,org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor,org.apache.hadoop.hbase.security.access.ReadOnlyClusterEnabler,org.apache.hadoop.hbase.rsgroup.RSGroupAdminEndpoint	This coprocessor is used to handle Master-level operations after the HIndex function is enabled, for example, creating an index meta table, adding an index, and deleting an index, a table, and index metadata.
RegionServer > RegionServer	hbase.coprocessor.regionserver.classes	org.apache.hadoop.hbase.hindex.server.regionserver.HIndexRegionServerCoprocessor,org.apache.hadoop.hbase.JMXListener,org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor	This coprocessor is used to handle the operations that the Master delivers to RegionServer after the HIndex function is enabled.

Navigation Path	Parameter	Default Value	Description
	hbase.coprocessor.region.classes	org.apache.hadoop.hbase.hindex.server.regionserver.HIndexRegionCoprocessor,org.apache.hadoop.hbase.security.token.TokenProvider,com.xxx.hadoop.hbase.backup.services.RecoveryCoprocessor,org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor,org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint,org.apache.hadoop.hbase.security.access.ReadOnlyClusterEnabler,org.apache.hadoop.hbase.coprocessor.MetaTableMetrics	This coprocessor is used to operate data in the Region after the HIndex function is enabled.

NOTE

- The preceding default values need to be configured after the HBase HIndex function is enabled. In MRS clusters that support the HBase HIndex function, the values have been configured by default.
- Ensure that the **master** parameter is configured on HMaster and the **region** and **regionserver** parameters are configured on RegionServer.

Interfaces Related to HBase Local Secondary Indexes

The APIs that use HIndex are in the **org.apache.hadoop.hbase.hindex.client.HIndexAdmin** class. The following table describes the related APIs.

Operation	API	Description	Precautions
Add an index.	addIndices()	Add an index to a table without data. Calling this API will add the specified index to a table but skips index data generation. Therefore, after this operation, the index cannot be used for the scanning and filtering operations. This API applies to scenarios where users want to add indexes in batches to tables that have a large amount of pre-existing user data. The specific operation is to use external tools such as the TableIndexer tool to build index data.	• An index cannot be modified once it is added. To modify the index, you need to delete the old index and then create a new one. • Do not create two indexes on the same column with different index names. Otherwise, storage and processing resources will be wasted. • Indexes cannot be added to a system table. • The append and increment operations are not supported when data is put into the index column. • If any fault occurs on the client except **DoNotRetryIOException**, you need to try again. • Index column families are selected from existing column families in the data table based on the following priorities in descending order: d, #, @, $, %, #0, @0, $0, %0, #1, @1 ... up to #255, @255, $255, and %255 When an index is created, the system checks whether the preceding column
	addIndicesWithData()	Add an index to a table with data. This API is used to add the specified index to the table and create index data for the existing user data. Alternatively, the API can be called to generate an index and then generate index data when the user data is being stored. Therefore, after this operation, the index can be used for the scanning and filtering operations immediately.	

Operation	API	Description	Precautions
			families exist in the table according to the preceding priority order. If the column families do not exist, the system sets the first column family that does not exist as the index column family. For example: – If the data table contains only the **d** column family, the index column family is **#** by default. – If the **d** and **#** column families already exist in the data table, the default index column family is **@**. – If the **d**, **#**, and **$** column families already exist in the data table, the index column family is **@** by default. • You can use the HIndex TableIndexer tool to add indexes without building index data.

Operation	API	Description	Precautions
Delete an index.	dropIndices()	This API is used to delete an index only. It deletes the specified index from a table but skips the corresponding index data. After this operation, the index cannot be used for the scanning and filtering operations. The cluster automatically deletes old index data during major compaction. This API applies to scenarios where a table contains a large amount of index data and **dropIndicesWithData()** is unavailable. In addition, you can use the TableIndexer tool to delete indexes and index data.	• An index can be disabled when it is in the **ACTIVE**, **INACTIVE**, or **DROPPING** state. • If you use **dropIndices()** to delete an index, ensure that the index data has been deleted before the index is added to the table with the same index name (that is, major compaction has been completed). • If you delete an index, the following information will also be deleted: – A column family with an index – Any one of column families in a combination index • Indexes and index data can be deleted together using the HIndex TableIndexer tool.
	dropIndicesWithData()	Delete index data. This API deletes the specified index and all index data corresponding to the index in a user table. After this operation, the index is completely deleted from the table and is no longer used for the scanning and filtering operations.	

Operation	API	Description	Precautions
Enable/ Disable an index.	disableIndices()	This API disables all indexes specified by a user so that they are no longer used for the scanning and filtering operations.	• An index can be enabled when the index is in the **ACTIVE**, **INACTIVE**, or **BUILDING** state. • An index can be disabled when the index is in the **ACTIVE** or **INACTIVE** state. • Before disabling an index, ensure that the index data is consistent with the user data. If no new data is added to the table when the index is disabled, the index data is consistent with the user data. • When enabling an index, you can use the TableIndexer tool to build index data to ensure data consistency.
	enableIndices()	This API enables all indexes specified by a user so that they can be used for the scanning and filtering operations.	
View the created index.	listIndices()	This API is used to list all indexes of a specified table.	N/A

Querying Data Based on HBase Local Secondary Indexes

You can use a filter to query data in a user table with an index. The query result of a user table with a single or combination index is the same as that of a table without an index, but the table with an index provides higher data query performance than the table without an index.

The index usage rules are as follows:

- Scenario 1: A single index is created for one or more columns.
  - When this column is used for AND or OR query filtering, an index can improve query performance.

    Example: Filter_Condition(IndexCol1)AND / OR Filter_Condition(IndexCol2)
  - When you use **Index Column** AND **Non-Index Column** for filtering in the query, the index can improve query performance.

Example: Filter_Condition(IndexCol1)AND Filter_Condition(IndexCol2)AND Filter_Condition(NonIndexCol1)

- When you use **Index Column** OR **Non-Index Column** for filtering in the query but do not use an index, query performance will not be improved.

  Example: Filter_Condition(IndexCol1)AND / OR Filter_Condition(IndexCol2) OR Filter_Condition(NonIndexCol1)

- Scenario 2: A combination index is created for multiple columns.
  - When the columns to be queried are all or part of the combination index and have the same order as the combination index, using the index improves query performance.

    For example, create a combination index for C1, C2, and C3.

    - The index takes effect in the following situations:

      Filter_Condition(IndexCol1)AND Filter_Condition(IndexCol2)AND Filter_Condition(IndexCol3)

      Filter_Condition(IndexCol1)AND Filter_Condition(IndexCol2)

      FILTER_CONDITION(IndexCol1)

    - The index does not take effect in the following situations:

      Filter_Condition(IndexCol2)AND Filter_Condition(IndexCol3)

      Filter_Condition(IndexCol1)AND Filter_Condition(IndexCol3)

      FILTER_CONDITION(IndexCol2)

      FILTER_CONDITION(IndexCol3)

  - When you use **Index Column** AND **Non-Index Column** for filtering in the query, the index can improve query performance.

    Examples:

    Filter_Condition(IndexCol1)AND Filter_Condition(NonIndexCol1)

    Filter_Condition(IndexCol1)AND Filter_Condition(IndexCol2)AND Filter_Condition(NonIndexCol1)

  - When you use **Index Column** OR **Non-Index Column** for filtering in the query but do not use an index, query performance will not be improved.

    Examples:

    Filter_Condition(IndexCol1)OR Filter_Condition(NonIndexCol1)

    (Filter_Condition(IndexCol1)AND Filter_Condition(IndexCol2))OR(Filter_Condition(NonIndexCol1))

  - When multiple columns are used for query, you can specify a value range for only the last column in the combination index and set other columns to specified values

    For example, create a combination index for C1, C2, and C3. In a range query, only the value range of C3 can be set. The filter criteria are "C1 = *XXX*, C2 = *XXX*, and C3 = Value range."

HBase Local Secondary Index Query Policy Selection

Use **SingleColumnValueFilter** or **SingleColumnRangeFilter**. It will provide the definite value **column_family:qualifierpair** (called **col1**) in filter criteria.

If **col1** is the first index column in the table, any index in the table can be a candidate index used during the query. The following provides an example:

If there is an index on **col1**, the index can be used as a candidate index because **col1** is the first and the only column of the index. If there is another index on **col1** and **col2**, you can consider this index as a candidate index because **col1** is the first column in the index list. However, if there is an index on **col2** and **col1**, this index cannot be used as a candidate index because the first column in the index list is not **col1**.

The most suitable method to use the index now is that when there are multiple candidate indexes, select the most suitable index for scanning data.

You can use the following solutions to learn how to select the best index policy.

- It is better to fully match.

  Scenario: There are two indexes available, one for **col1&col2** and the other for **col1**.

  In this scenario, the second index is better than the first one, because it scans less index data.

- If there are multiple candidate multi-column indexes, select an index with fewer index columns.

  Scenario: There are two indexes available, one for **col1&col2** and the other for **col1&col2&col3**.

  In this case, you had better use the index on **col1&col2**, because it scans less index data.

NOTE

- During a query based on an index, the index state must be **ACTIVE**. You can call the **listIndices()** API to view the index state.
- To query the correct data based on the index, ensure the consistency between index data and user data.
- Run the following command to perform a complex query on the HBase shell client (assuming that an index has been created for the specified column):

  scan 'tablename', {FILTER => "SingleColumnValueFilter(family, qualifier, compareOp, comparator, filterIfMissing, latestVersionOnly)"}

  Example: **scan 'test', {FILTER => "SingleColumnValueFilter('info', 'age', =, 'binary:26', true, true)"}**

  In the preceding scenario, if you want to save the row where no column is found in the result, you should not create any index in any such column, because if the column to be queried does not exist, the row will be filtered out when SCVF is used to scan the index columns. When the SCVF whose **filterIfMissingset** is **false** (default value) scans non-index columns, rows where no column is queried will also be returned in the result. Therefore, to avoid inconsistent query results, you are advised to set **filterIfMissing** to **true** after creating SCVF for the index column.
- Run the following command on the **hbase shell** client to view the index data created for user data:

  scan 'tablename', {ATTRIBUTES => {'FETCH_INDEX_DATA' => 'true'}}

9.6.2.2 Loading Index Data in Batches and Generating Local Secondary Indexes

Scenarios

HBase provides the ImportTsv&LoadIncremental tool to load user data in batches. HBase also provides the HIndexImportTsv tool to load both the user data and index data in batches. HIndexImportTsv inherits all functions of the HBase batch data loading tool ImportTsv. If a table is not created before the HIndexImportTsv tool is executed, an index will be created when the table is created, and index data is generated when user data is generated.

Procedure

1. Run the following commands to import data to HDFS:

   hdfs dfs -mkdir <*inputdir*>

   hdfs dfs -put <local_data_file> <*inputdir*>

   For example, define the data file **data.txt** as follows:

   ```
   12005000201,Zhang San,Male,19,City a, Province a
   12005000202,Li Wanting,Female,23,City b, Province b
   12005000203,Wang Ming,Male,26,City c, Province c
   12005000204,Li Gang,Male,18,City d, Province d
   12005000205,Zhao Enru,Female,21,City e, Province e
   12005000206,Chen Long,Male,32,City f, Province f
   12005000207,Zhou Wei,Female,29,City g, Province g
   12005000208,Yang Yiwen,Female,30,City h, Province h
   12005000209,Xu Bing,Male,26,City i, Province i
   12005000210,Xiao Kai,Male,25,City j, Province j
   ```

   Run the following commands:

   hdfs dfs -mkdir /datadirImport

   hdfs dfs -put data.txt /datadirImport

2. Go to HBase shell and run the following command to create the **bulkTable** table:

   create 'bulkTable', {NAME => 'info',COMPRESSION => 'SNAPPY', DATA_BLOCK_ENCODING => 'FAST_DIFF'},{NAME=>'address'}

   After the execution is complete, exit the HBase shell.

3. Run the following commands to generate an HFile file (StoreFiles):

   hbase org.apache.hadoop.hbase.hindex.mapreduce.HIndexImportTsv -Dimporttsv.separator=*<separator>*

   -Dimporttsv.bulk.output=*</path/for/output>* **-Dindexspecs.to.add=**<indexspecs> **-Dimporttsv.columns=**<columns> tableName *<inputdir>*

   - **-Dimport.separator**: indicates a separator, for example, **-Dimport.separator=','**.
   - **-Dimport.bulk.output=</path/for/output>**: indicates the output path of the execution result. You need to specify a path that does not exist.
   - **<columns>**: Indicates the mapping of the imported data in a table, for example, **-Dimporttsv.columns=HBASE_ROW_KEY,info:name,info:gender,info:age,address:city,address:province**.

- **<tablename>**: Indicates the name of a table to be operated.
- **<inputdir>**: Indicates the directory where data is loaded in batches.
- **-Dindexspecs.to.add=<indexspecs>**: Indicates the mapping between an index name and a column, for example, **-Dindexspecs.to.add='index_bulk=>info:[age->String]'**. The index composition can be represented as follows:

  indexNameN=>familyN :[columnQualifierN-> columnQualifierDataType], [columnQualifierM-> columnQualifierDataType];familyM: [columnQualifierO-> columnQualifierDataType]# indexNameN=> familyM: [columnQualifierO-> columnQualifierDataType]

  Column qualifiers are separated by commas (,).

  Example: "index1 => f1:[c1-> String],[c2-> String]"

  Column families are separated by semicolons (;).

  Example: "index1 => f1:[c1-> String],[c2-> String]; f2:[c3-> Long]"

  Multiple indexes are separated by pound keys (#).

  Example: "index1 => f1:[c1-> String],[c2-> String]; f2:[c3-> Long]#index2 => f2:[c3-> Long]"

  The following data types are supported by columns.

  Available data types are as follows: STRING, INTEGER, FLOAT, LONG, DOUBLE, SHORT, BYTE, CHAR

  NOTE

  Data types can also be transferred in lowercase.

For example, run the following command:

hbase org.apache.hadoop.hbase.hindex.mapreduce.HIndexImportTsv -Dimporttsv.separator=',' -Dimporttsv.bulk.output=/dataOutput -Dindexspecs.to.add='index_bulk=>info:[age->String]' -Dimporttsv.columns=HBASE_ROW_KEY,info:name,info:gender,info:age,address:city,address:province bulkTable /datadirImport/data.txt

Command output:

```
[root@shap000000406 opt]# hbase org.apache.hadoop.hbase.hindex.mapreduce.HIndexImportTsv -Dimporttsv.separator=',' -Dimporttsv.bulk.output=/dataOutput -Dindexspecs.to.add='index_bulk=>info:[age->String]' -Dimporttsv.columns=HBASE_ROW_KEY,info:name,info:gender,info:age,address:city,address:province bulkTable /datadirImport/data.txt
2018-05-08 21:29:16,059 INFO  [main] mapreduce.HFileOutputFormat2: Incremental table bulkTable output configured.
2018-05-08 21:29:16,069 INFO  [main] client.ConnectionManager$HConnectionImplementation: Closing master protocol: MasterService
2018-05-08 21:29:16,069 INFO  [main] client.ConnectionManager$HConnectionImplementation: Closing zookeeper sessionid=0x80007c2cb4fd5b4d
2018-05-08 21:29:16,072 INFO  [main] zookeeper.ZooKeeper: Session: 0x80007c2cb4fd5b4d closed
2018-05-08 21:29:16,072 INFO  [main-EventThread] zookeeper.ClientCnxn: EventThread shut down for session: 0x80007c2cb4fd5b4d
2018-05-08 21:29:16,379 INFO  [main] client.ConfiguredRMFailoverProxyProvider: Failing over to 147
2018-05-08 21:29:17,328 INFO  [main] input.FileInputFormat: Total input files to process : 1
2018-05-08 21:29:17,413 INFO  [main] mapreduce.JobSubmitter: number of splits:1
2018-05-08 21:29:17,430 INFO  [main] Configuration.deprecation: io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-05-08 21:29:17,687 INFO  [main] mapreduce.JobSubmitter: Submitting tokens for job: job_1525338489458_0002
2018-05-08 21:29:18,100 INFO  [main] impl.YarnClientImpl: Submitted application application_1525338489458_0002
2018-05-08 21:29:18,136 INFO  [main] mapreduce.Job: The url to track the job: http://
```



```
shap000000407:8088/proxy/application_1525338489458_0002/
2018-05-08 21:29:18,136 INFO  [main] mapreduce.Job: Running job: job_1525338489458_0002
2018-05-08 21:29:28,248 INFO  [main] mapreduce.Job: Job job_1525338489458_0002 running in uber
mode : false
2018-05-08 21:29:28,249 INFO  [main] mapreduce.Job:  map 0% reduce 0%
2018-05-08 21:29:38,344 INFO  [main] mapreduce.Job:  map 100% reduce 0%
2018-05-08 21:29:51,421 INFO  [main] mapreduce.Job:  map 100% reduce 100%
2018-05-08 21:29:51,428 INFO  [main] mapreduce.Job: Job job_1525338489458_0002 completed
successfully
2018-05-08 21:29:51,523 INFO  [main] mapreduce.Job: Counters: 50
```

4. Run the following command to import the generated HFile to HBase:

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *</path/for/output> <tablename>*

   For example, run the following command:

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles / dataOutput bulkTable

   Command output:

```
[root@shap000000406 opt]# hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /
dataOutput bulkTable
2018-05-08 21:30:01,398 WARN  [main] mapreduce.LoadIncrementalHFiles: Skipping non-directory
hdfs://hacluster/dataOutput/_SUCCESS
2018-05-08 21:30:02,006 INFO  [LoadIncrementalHFiles-0] hfile.CacheConfig: Created cacheConfig:
CacheConfig:disabled
2018-05-08 21:30:02,006 INFO  [LoadIncrementalHFiles-2] hfile.CacheConfig: Created cacheConfig:
CacheConfig:disabled
2018-05-08 21:30:02,006 INFO  [LoadIncrementalHFiles-1] hfile.CacheConfig: Created cacheConfig:
CacheConfig:disabled
2018-05-08 21:30:02,085 INFO  [LoadIncrementalHFiles-2] compress.CodecPool: Got brand-new
decompressor [.snappy]
2018-05-08 21:30:02,120 INFO  [LoadIncrementalHFiles-0] mapreduce.LoadIncrementalHFiles: Trying
to load hfile=hdfs://hacluster/dataOutput/address/042426c252f74e859858c7877b95e510
first=12005000201 last=12005000210
2018-05-08 21:30:02,120 INFO  [LoadIncrementalHFiles-2] mapreduce.LoadIncrementalHFiles: Trying
to load hfile=hdfs://hacluster/dataOutput/info/f3995920ae0247a88182f637aa031c49
first=12005000201 last=12005000210
2018-05-08 21:30:02,128 INFO  [LoadIncrementalHFiles-1] mapreduce.LoadIncrementalHFiles: Trying
to load hfile=hdfs://hacluster/dataOutput/d/c53b252248af42779f29442ab84f86b8 first=\x00index_bulk
\x00\x00\x00\x00\x00\x00\x00\x0018\x00\x0012005000204 last=\x00index_bulk
\x00\x00\x00\x00\x00\x00\x00\x0032\x00\x0012005000206
2018-05-08 21:30:02,231 INFO  [main] client.ConnectionManager$HConnectionImplementation:
Closing master protocol: MasterService
2018-05-08 21:30:02,231 INFO  [main] client.ConnectionManager$HConnectionImplementation:
Closing zookeeper sessionid=0x81007c2cf0f55cc5
2018-05-08 21:30:02,235 INFO  [main] zookeeper.ZooKeeper: Session: 0x81007c2cf0f55cc5 closed
2018-05-08 21:30:02,235 INFO  [main-EventThread] zookeeper.ClientCnxn: EventThread shut down
for session: 0x81007c2cf0f55cc5
```

9.6.2.3 Using TableIndexer to Generate a Local HBase Secondary Index

Scenarios

To quickly create indexes for user data, HBase provides the TableIndexer tool for you to create, add, and delete indexes using MapReduce functions. The application scenarios are as follows:

- You want to add an index for a specified column in a table where a large amount of data exists. However, if you use the **addIndicesWithData()** API to add an index, index data corresponding to the related user data will be generated, which is time-consuming. If you use **addIndices()** to create an index, index data corresponding to user data will not be generated. Therefore,

to create index data for user data, you can use the TableIndexer tool to create an index.

- If the index data is inconsistent with the user data, the tool can be used to rebuild index data.

  If you temporarily disable the index, put new data to the disabled index column, and then directly enable the index from the disabled state, index data and user data may be inconsistent. Therefore, you must rebuild all index data before using it again.
- You can use the TableIndexer tool to completely delete a large amount of existing index data from a user table.
- For user tables that do not have indexes, this tool allows you to add and build indexes at the same time.

How to Use

- **Adding a new index to a user table**

  The command is as follows:

  hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=tablename -Dindexspecs.to.add='idx_0=>cf_0:[q_0->string],[q_1];cf_1:[q_2],[q_3]#*idx_1*=>cf_1:[q_4]'

  The following parameters are required.

  - **tablename.to.index**: Indicates the name of a table for which an index is created.
  - **indexspecs.to.add**: Indicates the mapping between the index name and the column in the corresponding user table.
  - **scan.caching** (optional): Contains an integer value, indicating the number of cached rows to be transmitted to the scanner during data table scanning.

  The parameters in the preceding command are described as follows:

  - **idx_1**: Indicates an index name.
  - **cf_0**: Indicates the name of a column family.
  - **q_0**: Indicates the name of a column.
  - **string**: Indicates a data type. The parameter value can be STRING, INTEGER, FLOAT, LONG, DOUBLE, SHORT, BYTE, or CHAR.

  NOTE

  - The pound key (#) is used to separate indexes. The semicolon (;) is used to separate column families. The comma (,) is used to separate column qualifiers.
  - The column name and its data type must be included in '[]'.
  - Column names and their data types are separated by '->'.
  - If the data type of a specific column is not specified, the default data type (string) is used.
  - If **scan.caching** is not configured, the default value **1000** is used.
  - The user table must exist.
  - The index specified in the table must not exist.
  - If a column family named **d** exists in the user table, you must use the TableIndexer tool to build index data.

After the preceding command is executed, the specified index is added to the table and is in INACTIVE state. This behavior is similar to the **addIndices()** API.

- **Creating index data for existing indexes in a user table**

  The command is as follows:

  hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=tablename -Dindexnames.to.build='idx_0#idx_1'

  The following parameters are required.

  – **tablename.to.index**: Indicates the name of a table for which an index is created.

  – **indexspecs.to.build**: Indicates an index name.

  – **scan.caching** (optional): Contains an integer value, indicating the number of cached rows to be transmitted to the scanner during data table scanning.

  The parameters in the preceding command are described as follows:

  – **idx_1**: Indicates an index name.

  NOTE

  - The pound key (#) is used to separate index names.
  - If **scan.caching** is not configured, the default value **1000** is used.
  - The user table must exist.

  After the preceding command is executed, the specified index is set to the ACTIVE state. Users can use them when scanning data.

- **Deleting the existing indexes and their data from a user table**

  The command is as follows:

  hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=tablename -Dindexnames.to.drop='idx_0#idx_1'

  The following parameters are required.

  – **tablename.to.index**: Indicates the name of a table for which an index is created.

  – **indexnames.to.drop**: Indicates the name of the index that should be deleted with its data (must exist in the table).

  – **scan.caching** (optional): Contains an integer value, indicating the number of cached rows to be transmitted to the scanner during data table scanning.

  The parameters in the preceding command are described as follows:

  – **idx_1**: Indicates an index name.

  NOTE

  - The pound key (#) is used to separate index names.
  - If **scan.caching** is not configured, the default value **1000** is used.
  - The user table must exist.

  After the preceding command is executed, the specified index is deleted from the table.

- **Adding new indexes to user tables and building data based on existing data**

The command is as follows:

hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=tablename -Dindexspecs.to.add='idx_0 => cf_0:[q_0-> string],[q_1];cf_1:[q_2],[q_3]#idx_1 => cf_1:[q_4]' -Dindexnames.to.build='idx_0'

NOTE

- The parameters are the same as the previous ones.
- The user table must exist.
- The indexes specified in **indexspecs.to.add** must not exist in the table.
- The index names specified in **indexnames.to.build** must exist in the table or be part of the value of **indexspecs.to.add**.

After the preceding command is executed, all indexes specified in **indexspecs.to.add** will be added to this table, and index data will be built for all specified indexes using **indexnames.to.build**.

9.6.3 Improving HBase BulkLoad Data Migration

9.6.3.1 Importing HBase Data in Batches Using BulkLoad

Scenario

You can run commands to import data to HBase in batches and create indexes.

You can define multiple methods in **configuration.xml** for importing data in batches. You do not need to create indexes during data importing.

NOTE

- The column name consists of letters, digits, and underscores (_) and cannot contain any special characters.
- If the MapReduce job fails to be executed, rectify the fault by following the instructions provided in **Why Physical Memory Overflow Occurs If a MapReduce Task Fails?**.
- The data sources supported by BulkLoad are text files with separators.
- The client has been installed. For example, the installation directory is **/opt/hadoopclient**. The client directory in the following operations is only an example. Change it to the actual installation directory.
- If secondary indexes are created when data is imported in batches, pay attention to the following points:
  - If the column type is set to string, the string length cannot be set. For example, <column index="1" type="string" **length="1"** >COLOUMN_1</column> is not supported.
  - If the column type is set to date, the date format cannot be set. For example, <column index="13" type="date" **format="yyyy-MM-dd hh:mm:ss"**>COLOUMN_13</column> is not supported.
  - Secondary indexes cannot be created for combined columns.

Importing HBase Data in Batches Using the BulkLoad Tool

Step 1 Log in to the node where the client is installed as the client installation user.

Step 2 Run the following command to go to the client directory:

cd /opt/hadoopclient

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Run the following command to authenticate the current user if Kerberos authentication is enabled for the current cluster. The current user must have the permissions to submit YARN jobs, create and write HBase tables, and use HDFS.

kinit *Component service user*

Run the following command to set the Hadoop username if Kerberos authentication is not enabled for the current cluster:

export HADOOP_USER_NAME=hbase

Step 5 Run the following commands to import data to HDFS:

hdfs dfs -mkdir <*inputdir*>

hdfs dfs -put <local_data_file> <*inputdir*>

For example, define data file **data.txt** as follows:

```
001,Hadoop,citya
002,HBaseFS,cityb
003,HBase,cityc
004,Hive,cityd
005,Streaming,citye
006,MapReduce,cityf
007,Kerberos,cityg
008,LdapServer,cityh
```

Run the following command:

hdfs dfs -mkdir /datadirImport

hdfs dfs -put data.txt /datadirImport

Step 6 Go to **hbase shell**, create the table **ImportTable** and file **configuration.xml** (this file can be edited by referring to the template file in **/opt/client/HBase/hbase/conf/import.xml.template**).

For example, run the following command to create the table:

create 'ImportTable', {NAME => 'f1',COMPRESSION => 'SNAPPY', DATA_BLOCK_ENCODING => 'FAST_DIFF'},{NAME=>'f2'}

For example, the content of the customized import template file **configuration.xml** is as follows:

NOTE

- The value of **column_num** must be consistent with the number of columns in the data file.
- The specified family must correspond to the column family of the table.
- The following parameters need to be set only when secondary indexes are created during batch data import. The first letter of the index type must be capitalized, for example, **type="String"**. In the following snippets, **length="30"** indicates that the value of the index column **H_ID** cannot exceed 30 characters:

```
<indices>
   <index name="IDX1">
      <index_column family="f1">
         <qualifier type="String" length="30">H_ID</qualifier>
      </index_column>
   </index>
</indices>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<configuration>
    <import id="first" column_num="3">
        <columns>
            <column index="1" type="int">SMS_ID</column>
            <column index="2" type="string">SMS_NAME</column>
            <column index="3" type="string">SMS_ADDRESS</column>
        </columns>

        <rowkey>
            SMS_ID+'_'+substring(SMS_NAME,1,4)+'_'+reverse(SMS_ADDRESS)
        </rowkey>

        <qualifiers>
            <normal family="f1">
                <qualifier column="SMS_ID">H_ID</qualifier>
                <qualifier column="SMS_NAME">H_NAME</qualifier>
                <qualifier column="SMS_ADDRESS">H_ADDRESS</qualifier>
            </normal>

            <!-- Define composite columns -->
            <composite family="f2">
                <qualifier class="com.huawei.H_COMBINE_1">H_COMBINE_1</qualifier>
                <columns>
                    <column>SMS_ADDRESS</column>
                    <column>SMS_NAME</column>
                </columns>
            </composite>

        </qualifiers>

          <indices>
            <index name="IDX1">
                <index_column family="f1">
                    <qualifier type="String" length="30">H_ID</qualifier>
                </index_column>
            </index>
        </indices>

        <badlines>SMS_ID &lt; 7000 &amp;amp;&amp;amp; SMS_NAME == 'HBase'</badlines>
    </import>
</configuration>
```

Step 7 Run the following commands to generate an HFile file:

hbase com.huawei.hadoop.hbase.tools.bulkload.ImportData -Dimport.skip.bad.lines=*true* **-Dimport.separator**=*<separator>* **-Dimport.bad.lines.output**=*</path/badlines/output>* **-Dimport.hfile.output=***</path/for/output> <configuration xmlfile> <tablename> <inputdir>*

- **-Dimport.skip.bad.lines**: If this parameter is set to **false**, the command execution stops when an inapplicable row occurs. If this parameter is set to **true**, when an inapplicable row occurs, this row is skipped and the command execution continues. If no inapplicable row is defined in **configuration.xml**, this parameter does not need to be added.
- **-Dimport.separator**: indicates a separator, for example, **-Dimport.separator=','**.
- **-Dimport.bad.lines.output**=*</path/badlines/output>*: indicates the output path of the inapplicable data row. If no inapplicable data row is defined in **configuration.xml**, this parameter does not need to be added.
- **-Dimport.hfile.output**=*</path/for/output>*: indicates the output path of the execution result.
- *<configuration xmlfile>*: points to the **configuration** file.
- *<tablename>*: indicates the name of a table to be operated.
- *<inputdir>*: data directory to be uploaded in batches.

For example, run the following command:

- **hbase com.huawei.hadoop.hbase.tools.bulkload.ImportData -Dimport.skip.bad.lines=true -Dimport.separator=',' -Dimport.bad.lines.output=/badline -Dimport.hfile.output=/hfile configuration.xml ImportTable /datadirImport**
- **hbase com.huawei.hadoop.hbase.tools.bulkload.IndexImportData -Dimport.skip.bad.lines=true -Dimport.separator=',' -Dimport.bad.lines.output=/badline -Dimport.hfile.output=/hfile configuration_index.xml IndexImportTable /datadirIndexImport**

NOTICE

- After transparent encryption is configured for HBase, when you run the bulkload command to generate an HFile, the HFile path specified by **-Dimport.hfile.output** must be a subdirectory in /*HBase root directory*/**extdata**, for example, **/hbase/extdata/bulkloadTmp/hfile**.
- After transparent encryption is configured for HBase, the HBase user who runs the bulkload command needs to be added to the **hadoop** user group of the corresponding cluster (the user group is c*<Cluster ID>*_hadoop for the cluster that is not the first installed on FusionInsight Manager, for example, **c2_hadoop**) and has the read permission on the encryption key of the HBase root directory.
- Check the permission on the **/tmp/hbase** directory and manually grant the write permission on the directory to the current user.

Step 8 Run the following command to import HFile to HBase:

- Importing data in batches:

  hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *</path/for/output> <tablename>*

- Create a secondary index when importing data in batches:

hbase org.apache.hadoop.hbase.hindex.mapreduce.HIndexLoadIncrementalHFiles *</path/for/output> <tablename>*

For example, run the following command:

- **hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /hfile ImportTable**
- **hbase org.apache.hadoop.hbase.hindex.mapreduce.HIndexLoadIncrementalHFiles /hfile IndexImportTable**

----End

9.6.3.2 Updating HBase Data in Batches Using BulkLoad

Scenario

Rows need to be updated in batches based on the row key naming rule, row key scope, field name, and field value.

Procedure

Run the following command to update the rows from row_start to row_stop and direct the output to **/output/destdir/**.

```
hbase com.huawei.hadoop.hbase.tools.bulkload.UpdateData
  -Dupdate.rowkey.start="row_start"
  -Dupdate.rowkey.stop="row_stop"
  -Dupdate.hfile.output=/user/output/
  -Dupdate.qualifier=f1:c1,f2
  -Dupdate.qualifier.new.value=0,a
  'table1'
```

- -Dupdate.rowkey.start="row_start": indicates that the start row number is row_start.
- -Dupdate.rowkey.stop="row_stop": indicates that the end row number is row_stop.
- -Dupdate.hfile.output=/user/output/: indicates that the output results are directed to **/user/output/**.

NOTICE

After transparent encryption is configured for HBase, see **Step 7** for precautions on batch updating.

Run the following command to load HFiles:

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *<path/for/output> <tablename>*

Precautions

1. During batch updating, the field value of the row that meets the requirements will be updated.

2. Batch updating cannot be performed on fields where indexes are created.
3. If you do not set the output file of the execution result, the default value is **/tmp/updatedata/***table name*.

9.6.3.3 Deleting HBase Data in Batches Using BulkLoad

Scenario

Rows need to be deleted in batches based on the row key naming rule, row key scope, field name, and field value.

Procedure

Run the following command to delete the rows from **row_start** to **row_stop** and direct the output to **/output/destdir/**.

```
hbase com.huawei.hadoop.hbase.tools.bulkload.DeleteData
  -Ddelete.rowkey.start="row_start"
  -Ddelete.rowkey.stop="row_stop"
  -Ddelete.hfile.output="/output/destdir/"
  -Ddelete.qualifier="cf1,cf0:vch,cf0:lng:1000"
  'table1'
```

- -Ddelete.rowkey.start="row_start": indicates that the start row number is row_start.
- -Ddelete.rowkey.stop="row_stop": indicates that the end row number is row_stop.
- -Ddelete.hfile.output="/output/destdir/": indicates that the output results are directed to **/output/destdir/**.
- -Ddelete.qualifier="cf1,cf0:vch,cf0:lng:1000": indicates that all columns in column family **cf1**, the **vch** column in column family **cf0**, and the column whose value is **1,000** in column **lng** in column family **cf0** are to be deleted.

NOTICE

If transparent encryption is configured for HBase, see **Step 7** for precautions on batch deletion.

Run the following command to load HFiles:

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles <*path/for/output*> <*tablename*>

Precautions

1. If indexes have been created for column qualifier, the field cannot be deleted in batches because batch deletion cannot be performed on fields where indexes are created.
2. If you do not set output data file **delete.hfile.output** of the execution result, the default value is **/tmp/deletedata/***table name*.

9.6.3.4 Counting Rows in an HBase Table Using BulkLoad

Scenario

Collect statistics on the number of rows that meet specific requirements based on the rowkey naming rule, rowkey scope, field name, and field value.

Procedure

Count the number of rows that meet the following requirements: The rowkey is in the scope from **row_start** to **row_stop**. The field name is **f3:age** and the field value is **25**. The first two characters of the rowkey is **mi**.

```
hbase com.huawei.hadoop.hbase.tools.bulkload.RowCounter -Dcounter.rowkey.start="row_start" -Dcounter.rowkey.stop="row_stop" -Dcounter.qualifier="f3:age:25" -Dcounter.rowkey.value="substring(0,2) == 'mi'" table1
```

- **-Dcounter.rowkey.start="row_start"**: indicates that the start rowkey is **row_start**.
- **-Dcounter.rowkey.stop="row_stop"**: indicates that the end rowkey is **row_stop**.
- **-Dcounter.qualifier="f3:age:25"**: indicates that the value of column **age** in column family **f3** is **25**.
- **-Dcounter.rowkey.value="substring(0,2) == 'mi'"**: indicates that the first two values of the rowkey is **mi**.

NOTE

If **row_start** and **row_stop** are specified, data whose values are greater than or equal to **row_start** and smaller than **row_stop** is counted.

9.6.3.5 BulkLoad Configuration File

This section describes how to use the BulkLoad tool to obtain required data.

Configure a User-Defined Combined Rowkey

When using the BulkLoad tool to import HBase data in batches, users can customize combined rowkeys. Combining rowkeys using BulkLoad is to process some column names in a customized manner using some rules and combine them to generate a new rowkey.

NOTE

The column name consists of letters, digits, and underscores and cannot contain any special characters.

Details about how to set **configuration.xml** to combine rowkeys are as follows. For example, combine column **SMS_ID**, the second to fourth characters of column **SMS_NAME**, and the reverse of column **SMS_SERAIL** (The parts are connected by underscores (_)).

```
<columns>
                <column index="1" type="int">SMS_ID</column>
                <column index="2" type="string">SMS_NAME</column>
                <column index="3" type="string">SMS_ADDRESS</column>
        </columns>
```



```
<rowkey>
    SMS_ID+'_'+substring(SMS_NAME,1,4)+'_'+reverse(SMS_ADDRESS)
</rowkey>
```

Table 9-6 Rowkey segment process functions

Function Prototype	Description	Example
format(data,"DataType")	Used to format string data.	For example, **format(data,"0.000")** is used to input data in "0.000" format.
converse(data,"yyyy-MM-dd","yyyyMMdd")	Used to convert the date format.	For example, **converse(data,"yyyy-MM-dd","yyyyMMdd")** is used change the date format from "yyyy-MM-dd" to "yyyyMMdd".
rand	Used to generate a random number. Only the int type is supported.	None
replace(data,"A","B")	Used to replace data.	**replace(data,"A","B")** is used to replace A with B.
reverse(data)	Used to reverse a character string.	For example, **reverse(ABC)** is used reverse "ABC" to "CBA".
substring(data,Length1,Length2), or substring(data,Length3)	Used to subtract a character string.	For example, **substring(data,1,5), or substring(data,3)** is used to subtract [1,5) or [3,data.length) from the data character string.
to_number("data")	Used to convert a character string into a numeric value. The Long type numeric value is supported.	For example, **to_number("123")** is used to convert "123" into 123. Note that "data" must be a numerical value.

Configuring Customized Rowkey Implementation

When the BulkLoad tool is used to import HBase data in batches, user-defined combined rowkeys are supported. You can compile RowKey implementation code and import combined RowKeys based on the code logic.

To configure a customized row key, perform the following steps:

Step 1 When you compile the implementation class of a custom RowKey, inherit the interface. The JAR file path of the interface is *Client installation directory*/**HBase/hbase/lib/hbase-it-bulk-load-*.jar**.

```
[com.huawei.hadoop.hbase.tools.bulkload.RowkeyHandlerInterface]
```

Interface implementation method:

```
byte[] getRowkeyBytes(String[] colsValues, RegulationDomain regulation)
```

Where

- **colsValues** indicates a collection of one original data row. Each element is a column.
- **regulation** indicates information about the configuration file to be imported. (Typically, this parameter is not used.)

Step 2 Compress the implementation class and its dependent package into a JAR file, save the file to any location on the node where the HBase client resides, and ensure that the user who executes the command has the permission to read and execute the JAR file.

Step 3 When you run the import command, add the following two configuration items:

-Dimport.rowkey.jar=*Full path of the JAR file in step 2*

-Dimport.rowkey.class=*Full class name of the user implementation class*

----End

Configuring Custom Combination Fields

You can combine fields in a customized manner using BulkLoad. Multiple columns are combined into one column in append mode.

NOTE

The column name consists of letters, digits, and underscores and cannot contain any special characters.

Details about how to combine fields into H_COMBINE_1 are as follows. For example, combine fields **SMS_SERIAL**, **SMS_ADDRESS**, and **SMS_SNAME** into **H_COMBINE_1**.

```
<!-- Define composite columns -->
                <composite family="f2">
                    <!-- define composited class name, and this class must not exists -->
                    <qualifier class="com.huawei.H_COMBINE_1">H_COMBINE_1</qualifier>
                    <columns>
                        <column>SMS_ADDRESS</column>
                        <column>SMS_NAME</column>
                    </columns>
                </composite>
```

Specifying the Field Data Type

HBase BulkLoad can read native data files, map fields in the data files to fields defined by HBase, and define data types of these fields.

You can define multiple methods in **configuration.xml** for importing data in batches.

NOTE

The column name consists of letters, digits, and underscores and cannot contain any special characters.

Details about how to specify the field data type are as follows. For example, specify data types for columns SMS_ID, SMS_NAME, SMS_ADDRESS, and SMS_SERIAL.

```
<columns>
        <column index="1" type="int">SMS_ID</column>
        <column index="2" type="string">SMS_NAME</column>
        <column index="3" type="string">SMS_ADDRESS</column>
 </columns>
```

NOTE

The supported data types include short, int, long, float, double, boolean, and string.

Defining Inapplicable Data Row

BulkLoad supports the function of defining inapplicable data rows. The inapplicable data rows are not stored in HBase. Instead, these data rows are stored in a specific file.

You can define multiple methods in **configuration.xml** for importing data in batches.

NOTE

The column name consists of letters, digits, and underscores and cannot contain any special characters.

Details about how to define inapplicable data rows are as follows:

```
<!-- Define bad line filter rule -->
<badlines>SMS_ID &lt; 7000 &amp;&amp; SMS_NAME == 'HBase'</badlines>
```

NOTE

SMS_ID < 7000 && SMS_NAME == 'HBase'

Table 9-7 lists the operators in <badlines> and corresponding parameter types.

Table 9-7 Operators and corresponding parameter types

Operator	Parameter Type
&&	The parameter type is Boolean.
&	The parameter type is integer.
|	The parameter type is integer.
^	The parameter type is integer.
/	The parameter type is digit.
==	The parameter type is string.
>=	The parameter type is digit.

Operator	Parameter Type
>	The parameter type is digit.
<<	The parameter type is integer.
<=	The parameter type is digit.
<	The parameter type is digit.
%	The parameter type is digit.
*	The parameter type is digit.
!=	The parameter type is string.
||	The parameter type is Boolean.
+	The parameter type is digit and string.
>>	The parameter type is integer.
-	The parameter type is string.
>>>	The parameter type is integer.

9.6.3.6 Configuring BulkLoad to Parse Customized Separators

Scenario

Phoenix provides CsvBulkloadTool, a batch data import tool. For details about related features, see **https://phoenix.apache.org/bulk_dataload.html**. This tool supports import of user-defined delimiters. Specifically, users can use any visible characters within the specified length as delimiters to import data files.

NOTE

This section applies only to MRS 3.2.0 or later.

Constraints

- User-defined delimiters cannot be an empty string.
- A user-defined delimiter can contain a maximum of 16 characters.

  NOTE

  A long delimiter affects parsing efficiency, slows down data import, reduces the proportion of valid data, and results in large files. Use short delimiters as possible.
- User-defined delimiters must be visible characters.

  NOTE

  A user-defined delimiter whitelist can be configured to avoid any injection issues possible. Currently, the following delimiters are supported: letters, numbers, and special characters (`~!@#$%^&*()\\-_=+\\[\\]{}\\\\|;:'\",<>./?).

- The start and end of a user-defined delimiter cannot be the same.

Description of New Parameters

The following two parameters are added based on the open source CsvBulkloadTool:

- **--multiple-delimiter(-md)**

  This parameter specifies the user-defined delimiter. If this parameter is specified, it takes effect preferentially and overwrites the **-d** parameter in the original command.

- **--multiple-delimiter-skip-check(-mdsc)**

  This parameter is used to skip the delimiter length and whitelist verification. It is not recommended.

Importing Data to HBase by User-defined Separator

Step 1 Upload the data file to the node where the client is deployed. For example, upload the **data.csv** file to the **/opt/test** directory on the target node. The delimiter is **|^[**. The file content is as follows:

```
0|^[lucy|^[81|^[city3|^[true|^[9.18914949740991|^[-22.18269890221688
1|^[zhangshan|^[46|^[city4|^[true|^[5.322036259193896|^[-38.48904063764235
2|^[james|^[11|^[city3|^[true|^[7.7681483995935885|^[-25.7800264590591
3|^[james|^[87|^[city2|^[false|^[81.60003424030911|^[-75.3247097149487
4|^[wangwu|^[35|^[city3|^[true|^[32.95621554646283|^[-53.02547887416576
5|^[james|^[71|^[city5|^[true|^[83.12680099627629|^[-10.747627215038714
6|^[wangwu|^[23|^[city1|^[false|^[91.50905273357168|^[-39.53314273786492
7|^[james|^[8|^[city2|^[false|^[12.530415667410132|^[-24.858112869804337
8|^[lucy|^[87|^[city1|^[false|^[39.39199423544571|^[-68.35869628818902
9|^[lucy|^[92|^[city3|^[true|^[33.1217894946113O6|^[-55.48365486185375
10|^[lucy|^[51|^[city5|^[true|^[32.883181700707965|^[-78.04791713353062
```

Step 2 Log in to the node where the client is installed as the client installation user.

Step 3 Run the following command to go to the client directory:

cd *Client installation directory*

Step 4 Run the following command to configure environment variables:

source bigdata_env

Step 5 Run the following command to authenticate the current user if Kerberos authentication is enabled for the current cluster. The current user must have the permissions to create HBase tables and operate HDFS.

kinit *Component service user*

Run the following command to set the Hadoop username if Kerberos authentication is not enabled for the current cluster:

export HADOOP_USER_NAME=hbase

Step 6 Run the following command to upload the data file **data.csv** in **Step 1** to an HDFS directory, for example, **/tmp**:

hdfs dfs -put /opt/test/data.csv /tmp

Step 7 Run the Phoenix client command.

sqlline.py

Step 8 Run the following command to create the **TEST** table:

CREATE TABLE TEST (ID INTEGER NOT NULL PRIMARY KEY, NAME VARCHAR, AGE INTEGER, ADDRESS VARCHAR, GENDER BOOLEAN, A DECIMAL, B DECIMAL) split on (1, 2, 3,4,5,6,7,8,9);

After the table is created, run the **!quit** command to exit the Phoenix CLI.

Step 9 Run the following import command:

hbase org.apache.phoenix.mapreduce.CsvBulkLoadTool -md '*User-defined delimiter***' -t** *Table name* **-i** *Data path*

For example, to import the **data.csv** file to the **TEST** table, run the following command:

hbase org.apache.phoenix.mapreduce.CsvBulkLoadTool -md '|^[' -t TEST -i /tmp/data.csv

Step 10 Run the following command to view data imported to the **TEST** table:

sqlline.py

SELECT * FROM TEST LIMIT 10;

```
No rows affected (2.307 seconds)
0: jdbc:phoenix:> SELECT * FROM TEST LIMIT 10;
+-----+-----------+-----+----------+--------+--------------------+----------------------+
| ID  |   NAME    | AGE | ADDRESS  | GENDER |         A          |          B           |
+-----+-----------+-----+----------+--------+--------------------+----------------------+
| 0   | lucy      | 81  | city3    | true   | 9.18914949740991   | -22.18269890221688   |
| 1   | zhangshan | 46  | city4    | true   | 5.322036259193896  | -38.48904063764235   |
| 2   | james     | 11  | city3    | true   | 7.7681483995935885 | -25.7800264590591    |
| 3   | james     | 87  | city2    | false  | 81.60003424030911  | -75.3247097149487    |
| 4   | wangwu    | 35  | city3    | true   | 32.95621554646283  | -53.02547887416576   |
| 5   | james     | 71  | city5    | true   | 83.12680099627629  | -10.747627215038714  |
| 6   | wangwu    | 23  | city1    | false  | 91.50905273357168  | -39.53314273786492   |
| 7   | james     | 8   | city2    | false  | 12.530415667410132 | -24.858112869804337  |
| 8   | lucy      | 87  | city1    | false  | 39.39199423544571  | -68.35869628818902   |
| 9   | lucy      | 92  | city3    | true   | 33.121789494611306 | -55.48365486185375   |
+-----+-----------+-----+----------+--------+--------------------+----------------------+
```

----End

9.6.4 Using the Spark BulkLoad Tool to Synchronize Data to HBase Tables

To quickly synchronize Hive or Spark table data to HBase tables, you can use the Spark BulkLoad tool. It also allows you to import full or incremental data in ORC/PAQUET format.

NOTE

Pay attention to the following when using the Spark BulkLoad tool:

- For details about data type conversion, see **Table 9-8**. The date type is converted to the string type before being stored in HBase. The number, string, and Boolean types are directly converted to byte arrays and stored in HBase. The system converts the byte arrays to the corresponding types during data parsing and check whether the values are null.
- Do not directly synchronize table data of the Struct, Map, and Seq types to HBase tables. These types cannot be converted to byte arrays, and will be converted to strings instead, which may fail to be restored.

This topic is available for MRS 3.5.0 and later versions only.

Table 9-8 Data type conversion relationship

Hive/Spark table	HBase Table	Parsing Mode
TINYINT	Byte	Returns the first value in byte[].
SMALLINT	Short	Bytes.toShort(byte[])
INT/INTEGER	Integer	Bytes.toInt(byte[])
BIGINT	Long	Bytes.toLong(byte[], int, int)
FLOAT	Float	Bytes.toFloat(byte[])
DOUBLE	Double	Bytes.toDouble(byte[])
DECIMAL/ NUMERIC	BigDecimal	Bytes.toBigDecimal(byte[])
TIMESTAMP	String	Bytes.toString(byte[])
DATE	String	Bytes.toString(byte[])
STRING	String	Bytes.toString(byte[])
VARCHAR	String	Bytes.toString(byte[])
CHAR	String	Bytes.toString(byte[])
BOOLEAN	Boolean	Bytes.toBoolean(byte[])
BINARY	byte[]	No need to parse.
ARRAY	String	Bytes.toString(byte[])
MAP	String	Bytes.toString(byte[])
STRUCT	String	Bytes.toString(byte[])

Prerequisites

- The Spark and Hive services have been installed in the cluster.

- The user who imports data must have the Spark permission (the SELECT permission of the source table), HBase permission (the RWXA permission of the HBase NameSpace), and HDFS permission (the read and write permission of the HFile output directory).
- If Kerberos authentication is enabled for the cluster (the cluster is in security mode), set the value of **spark.yarn.security.credentials.hbase.enabled** to **true** in the *Spark client installation directory*/**Spark/spark/conf/spark-defaults.conf configuration** file.

Spark BulkLoad Commands

The command format is as follows:

spark-submit --master yarn --deploy-mode cluster --jars *Client installation directory*/**HBase/hbase/lib/protobuf-java-2.5.0.jar**, *Client installation directory*/**HBase/hbase/conf/* --conf spark.yarn.user.classpath.first=true --class com.huawei.hadoop.hbase.tools.bulkload.SparkBulkLoadTool** *Client installation directory*/**HBase/hbase/lib/hbase-it-bulk-load-*.jar com.huawei.hadoop.hbase.tools.bulkload.SparkBulkLoadTool***[-cf <arg>] [-comp <arg>] [-enc <arg>]* **-op** *<arg>* **-rc** *<arg> [-rn <arg>] [-sp <arg>]* **-sql** *<arg> [-sr]* **-tb** *<arg>*

NOTE

- **--jars** specifies the path of the **protobuf-java-2.5.0.jar** file and the path of the HBase client configuration file. The HBase client configuration file is stored in *Client installation directory*/**HBase/hbase/conf**.
- The number of executors, memory, and CPU can be specified in the command for resource control. For example, the following parameters can be specified during command submission:

  --driver-memory=*20G* **--num-executors=***10* **--executor-memory=***4G* **--executor-cores=***2*

Other parameters that can be configured are as follows:

- **-sql,--export-sql** *<arg>*

  Sets the SQL statements for exporting data. When reading data from Hive/Spark tables, you can set this parameter to filter out data that does not need to be synchronized.

- **-rc,--rowkey-columns** *<arg>*

  Specifies the columns that compose the HBase Rowkey in the source table. If there are multiple columns, separate them with commas (,).

  NOTICE

  The Spark BulkLoad task will fail if the query is abnormal due to incorrect SQL statements, non-exist target data, or duplicate data. Ensure that the SQL statements are correct and the data combination corresponding to Rowkey fields is unique.

- **-sp,--rowkey-separator** *<arg>*

(Optional) Specifies the separator between field values when multiple column values are used as a rowkey. The default value is **#**. The values are concentrated to form the rowkey.

NOTICE

The separator can contain only one character. Avoid the characters used by the Rowkey field value to prevent failures in parsing column values. A composite rowkey (data rowkey) consists of multiple columns separated by the specified separator. To parse this rowkey, the separator must be located to split the rowkey splitting and converts data types. For example:

A rowkey consists of two columns separated by a number sign (#). **Table 9-9** shows the corresponding relationship. The code for parsing is as follows:

```
// Locate the separator.
int idx = Bytes.indexOf(row, "#".getBytes(StandardCharsets.UTF_8)[0]);
// Split the Rowkey and convert data types.
byte[] aBytes = ArrayUtils.subarray(row, 0, idx);
String aStr = Bytes.toString(aBytes);
byte[] bBytes = ArrayUtils.subarray(row, idx + 1, row.length);
Integer bInt = bBytes == null ? null : Bytes.toInt(bBytes);
```

Table 9-9 Composite rowkey example

Column A (String)	Column B (int)	Data Rowkey
a	1	a#1
b	null	b#

- **-tb,--table** *<arg>*

  Specifies the target HBase table. If the target table does not exist, sampling will be performed and the target table will be created.

- **-op,--output-path**

  Specifies the output path of HFiles. The exported HFiles are stored in a temporary directory in this directory and will be deleted after successful import.

  NOTE

  If HDFS federation is enabled, the HFile output path and the HBase to which data is to be imported must be in the same NameService.

  Table 9-10 shows an example of mounted HDFS directory. If the HBase service directory is mounted to NS1, the output path of the Spark Bulkload tool must be mounted to NS1. You can specify the output path to the **/tmpns1** directory.

Table 9-10 HDFS directory examples

Global Directory	Target NameService	Object Directory
/hbase	NS1	/hbase

Global Directory	Target NameService	Object Directory
/tmp	hacluster	/tmp
/tmpns1	NS1	/tmpns1

- **-rn,--region-nums** *<arg>*

  Specifies the number of target HBase regions. If the target table does not exist, this parameter value will be used to pre-partition the target table. The default value is 100.

  NOTE

  Evaluate the number of regions based on the amount of data to be exported from the source table. The estimation method is as follows:

  Size of the source table (three copies) x Decompression rate of the source table x HBase data expansion rate (estimated to 10)/Upper limit of a single region (usually 10 GB)/Compression and encoding compression rate

  Evaluate the number of regions. For example, if the source table is stored in ORC format and occupies 100 GB, the decompression expansion rate of the source table can be to 5. If sampled data is SNAPPY-compressed and FAST_DIFF-encoded in the target table, the compression rate can be 3. The minimum number of regions is: 100 x 5 x 10/10/3 ≈ 167. If you need to perform incremental data synchronization later, you can set the number of regions to 200.

- **-cf,--column-family** *<arg>*

  (Optional) Specifies the column family name of the target HBase table to which data is to be imported. If the column family does not exist, data synchronization will fail. If the target table does not exist, a table that contains this column family will be created in HBase. The default column family is **info**.

- **-comp,--compression** *<arg>*

  (Optional) Specifies the compression format of the target HBase table. Currently, SNAPPY, NONE, ZSTD, and GZ are supported. If the target table does not exist, a table using the specified compression format will be created in HBase. The default compression format is **SNAPPY**.

- **-enc,--block-encoding** *<arg>*

  (Optional) Specifies the data block encoding mode of the target HBase table. Currently, NONE, PREFIX, DIFF, FAST_DIFF, and ROW_INDEX_V1 are supported. If the target table does not exist, a table using the DATA BLOCK encoding mode will be created in HBase. The default value is **FAST_DIFF**.

- **-sr,--skip-store-rowcol**

  (Optional) Specifies whether to skip the columns corresponding to the Rowkey. By default, the Rowkey columns are redundantly stored in the HBase table. This parameter allows you to reduce the storage usage by skipping Rowkey parsing when it consists of multiple columns.

- **-sm,--sampling-multiple <arg>**

  (Optional) Specified the maximum number of files can be generated in a single region to improve the tool performance. There can be more ranges during sampling.

Note: A larger value indicates more generated HFiles, which increases the HBase compaction pressure. The value range is [1,10]. The default value is 1. You are advised to set this parameter based on actual resources.

Procedure

Step 1 Log in to the node where the client is installed as the client installation user.

Step 2 Go to the client directory.

cd *Client installation directory*

Step 3 Configure environment variables.

source bigdata_env

Step 4 If Kerberos authentication is enabled for the cluster, authenticate the user.

kinit *Component service user*

If Kerberos authentication is disabled for the cluster, set the Hadoop username.

export HADOOP_USER_NAME=hbase

Step 5 Go to the Spark client directory and synchronize data to the target HBase table.

cd Spark/spark/bin

For example, run the following command to synchronize all data in the **test.orc_table** table to the **test:orc_table** table of HBase, use **id+uuid** as the rowkey column, and set the output path to **/tmp/orc_table**:

spark-submit --master yarn --deploy-mode cluster --jars *Client installation directory*/**HBase/hbase/lib/protobuf-java-2.5.0.jar**, *Client installation directory*/**HBase/hbase/conf/* --conf spark.yarn.user.classpath.first=true --class com.huawei.hadoop.hbase.tools.bulkload.SparkBulkLoadTool** *Client installation directory*/**HBase/hbase/lib/hbase-it-bulk-load-*.jar -sql "select * from test.orc_table" -tb "test:orc_table" -rc "id,uuid" -op "/tmp/orc_table"**

----End

9.6.5 Configuring Hot-Cold Data Separate in HBase

9.6.5.1 Configuring Separate Storage for HBase Cold and Hot Data

In a big data storage scenario, HBase table data such as order data or monitoring data grows over time. As your business develops, such data can be of a large volume and rarely used. Companies may want to use cost-effective storage to store this type of data to reduce costs.

HBase separates cold data from hot data and stores them on different media. Cold data is stored in OBS and hot data is stored in HDFS, reducing storage costs.

This function is supported only by MRS 3.3.0 or later.

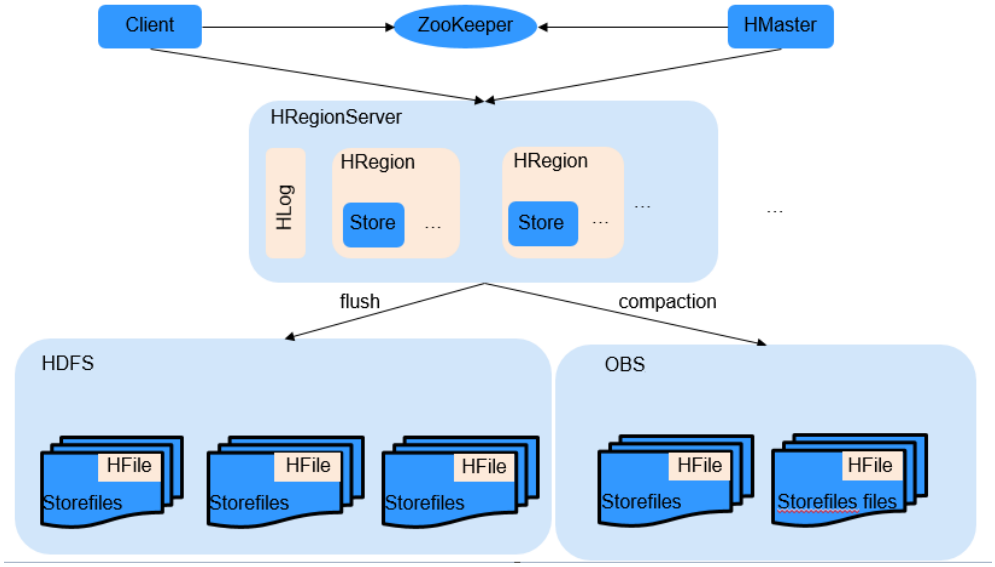
NOTE

- IOPS of data reading in OBS decreases. As a result, OBS is suitable for infrequent queries only.
- It is not a good choice to use OBS for a large number of concurrent read requests. Otherwise, exceptions may occur.

Principles

HBase supports separate cold and hot storage of data in the same table. After a user configures the time boundary between hot and cold data, HBase determines whether data is hot or cold based on the timestamp (ms) and the time boundary configured by the user. New data is stored in the hot storage and is gradually moved to the cold storage over time. You can change the time boundary for separating cold and hot data as you need. Data can be moved from the cold storage to the hot storage or vice versa.

Figure 9-5 HBase cold and hot separation principle



Configuring Separate Storage for HBase Cold and Hot Data

You can modify the HBase configuration on FusionInsight Manager to enable the cold and hot data separation feature so that cold data can be stored in OBS and hot data can be stored in HDFS.

Step 1 Interconnecting the Guardian Service with OBS. For details, see "Interconnecting the Guardian Service with OBS" section.

Step 2 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HBase**, and click **Configuration**. Search for and modify the following parameters:

- **fs.coldFS**: OBS file system name, for example, **obs://***OBS parallel file system name*
- **hbase.fs.hot.cold.enabled**: The default value is **false**. Set this parameter to **true**.

- **fs.obs.buffer.dir**: Set this parameter to the directory of the locally mounted data disk, for example, **/srv/BigData/data1/tmp/HBase/obs**.

Step 3 Click **Save**.

Step 4 Click **Dashboard** and click **More** > **Restart Service** to restart the HBase service. After the service is restarted, hot-cold data separation is enabled.

Step 5 Set the cold and hot boundary for the table data. For details, see **Cold-Hot Separation Commands**.

----End

9.6.5.2 Cold-Hot Separation Commands

The following content describes how to use the commands related to cold-hot separation, including shell commands and Java API commands.

Shell commands are executed on the HBase client. For details about how to install the client, see **Installing a Client**.

Setting the Hot and Cold Data Boundary of an HBase Table

- Shell
  - Create a table where data will be separately stored.

    create '*hot_cold_table*', {NAME=>'*f*', COLD_BOUNDARY=>'*86400*'}

    Required parameters are as follows:

    - **NAME** indicates the column family that requires cold-hot separation.
    - **COLD_BOUNDARY** indicates the time boundary (in seconds) for separating cold and hot data. For example, if **COLD_BOUNDARY** is set to **86400**, data that was written 86,400 seconds (one day) ago will be archived as cold data.

      NOTE

      The boundary time must be greater than the Major Compaction execution period. The default Major Compaction execution period is 7 days.
  - Disable cold-hot separation.

    alter '*hot_cold_table*', {NAME=>'*f*', COLD_BOUNDARY=>''}
  - Set cold and hot separation for an existing table or modify the cold and hot data boundary (unit: second) to convert between hot storage and cold storage. The following are some examples:
    - Convert hot data to cold data.

      1) Archive data that has been written to column **f** of the **hot_cold_table** table in more than one day (86400 seconds) to cold storage.

         alter '*hot_cold_table*', {NAME=>'*f*', COLD_BOUNDARY=>'*86400*'}

      2) Perform Major Compaction during off-peak hours to avoid affecting service performance.

         major_compact '*hot_cold_table*'

- Convert cold data to hot data.
  1) Change the value of **COLD_BOUNDARY** to **172800** to archive data that has been written to column **f** of the **hot_cold_table** table in more than one day and less than two days from the cold storage to the hot storage. Set this parameter as you need.

     alter '*hot_cold_table*', {NAME=>'*f*', COLD_BOUNDARY=>'*172800*'}
  2) Perform Major Compaction during off-peak hours to avoid affecting service performance.

     major_compact '*hot_cold_table*'

– Check whether cold-hot separation is enabled or whether the time boundary is successfully modified.

  desc '*hot_cold_table*'

```
Table hot_cold_table is ENABLED
hot_cold_table
COLUMN FAMILIES DESCRIPTION
{NAME => 'f', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING =>
'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER
=> 'ROW', IN_MEMORY => 'false', COMPRE
SSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536', METADATA =>
{'COLD_BOUNDARY' => '1200'}}
1 row(s)
Quota is disabled
Took 0.0339 seconds
```

- Java API
  – Create a table where data will be separately stored.

    COLD_BOUNDARY specifies the time boundary for separating cold and hot data. The time boundary is measured in seconds. In the following example, data that was written one day ago will be archived as cold data.

```
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("hot_cold_table");
HTableDescriptor descriptor = new HTableDescriptor(tableName);
HColumnDescriptor cf = new HColumnDescriptor("f");
cf.setValue(HColumnDescriptor.COLD_BOUNDARY, "86400");
descriptor.addFamily(cf);
admin.createTable(descriptor);
```

  – Disable cold-hot separation.

```
HTableDescriptor descriptor = admin.getTableDescriptor(tableName);
HColumnDescriptor cf = descriptor.getFamily("f".getBytes());
cf.setValue(HColumnDescriptor.COLD_BOUNDARY, null);
admin.modifyTable(tableName, descriptor);
```

  – Enable cold-hot separation for an existing table or change the time boundary.

    COLD_BOUNDARY specifies the time boundary for separating cold and hot data. The time boundary is measured in seconds. In the following example, data that was written one day ago will be archived as cold data.

```
HTableDescriptor descriptor = admin.getTableDescriptor(tableName);
HColumnDescriptor cf = descriptor.getFamily("f".getBytes());
cf.setValue(HColumnDescriptor.COLD_BOUNDARY, "86400");
admin.modifyTable(tableName, descriptor);
```

NOTE

You must perform a major compaction before you move the data between the cold storage and the hot storage.

Writing Data

You can write data to a table with separated cold and hot storage in the same way that you write data to a regular table. Newly written data is stored in hot storage (HDFS). If the storage duration of a data record exceeds the value specified by the **COLD_BOUNDARY** parameter, the system automatically moves the data to cold storage (OBS) during the compaction process.

- Insert a data record to a table.

  Run the **put** command to insert a data record to the specified table. Specify the table name, primary key, custom column, and value. The following is an example:

  put '*hot_cold_table*','*row1*','cf:*a*','*value1*'

  The following parameters are required in the command:

  - **hot_cold_table**: table name
  - **row1**: primary key
  - **cf: a**: custom column
  - **value1**: value to insert

Querying Data

Both cold data and hot data are in the same HBase table. You can query the data only on one table. You can configure **TimeRange** to specify the time range of the data you want to query. The system automatically determines whether the hot storage, cold storage, or both will be searched based on the specified time range. If the time range is not specified during the query, only cold storage will be searched. The throughput of reading cold data is lower than that of reading hot data.

NOTE

- The cold storage is used to archive data that is rarely accessed. If your cluster receives a large number of queries that hit cold data, you can check whether the time boundary (**COLD_BOUNDARY**) is set to an appropriate value. The query performance deteriorates if data that is frequently accessed are stored in the cold storage.
- If you update a field in a row that is stored in the cold storage, the field is moved to the hot storage after the update. When this row is hit by a query that carries the **HOT_ONLY** hint or has a time range that is configured to hit hot data, only the updated field in the hot storage is returned. If you want the system to return the entire row, you must delete the **HOT_ONLY** hint from the query statement or make sure that the specified time range covers the time period from when this row is inserted to when this row is last updated. It is recommended that you do not update data that is stored in the cold storage.

- Random queries with Get
  - Shell
    - Query data in cold storage without the **HOT_ONLY** hint.

      get '*hot_cold_table*', '*row1*'
    - Query data in hot storage with the **HOT_ONLY** hint.

      get '*hot_cold_table*', '*row1*', **{HOT_ONLY=>true}**

- Query data within a time range that is specified by **TimeRange**. The system determines whether the query hits cold or hot data based on the values of **TIMERANGE** and **COLD_BOUNDARY**.

  get '*hot_cold_table*', '*row1*', {TIMERANGE => [0, 1568203111265]}

  NOTE

  TimeRange specifies the query time range. The time in the range is a UNIX timestamp, which is the number of milliseconds that have elapsed since the Unix epoch.

– Java API

- Query data in cold storage without the **HOT_ONLY** hint.

```
Get get = new Get("row1".getBytes());
```

- Query data in hot storage with the **HOT_ONLY** hint.

```
Get get = new Get("row1".getBytes());
get.setAttribute(HBaseConstants.HOT_ONLY, Bytes.toBytes(true));
```

- Query data within a time range that is specified by **TimeRange**. The system determines whether the query hits cold or hot data based on the values of **TimeRange** and **COLD_BOUNDARY**.

```
Get get = new Get("row1".getBytes());
get.setTimeRange(0, 1568203111265)
```

  NOTE

  TimeRange specifies the query time range. The time in the range is a UNIX timestamp, which is the number of milliseconds that have elapsed since the Unix epoch.

- SCAN queries

– Shell

- Query data in cold storage without the **HOT_ONLY** hint.

  scan '*hot_cold_table*', {STARTROW =>'*row1*', STOPROW=>'*row9*'}

- Query data in hot storage with the **HOT_ONLY** hint.

  scan '*hot_cold_table*', {STARTROW =>'*row1*', STOPROW=>'*row9*', **HOT_ONLY=>true**}

- Query data within a time range that is specified by **TimeRange**. The system determines whether the query hits cold or hot data based on the values of **TIMERANGE** and **COLD_BOUNDARY**.

  scan '*hot_cold_table*', {STARTROW =>'*row1*', STOPROW=>'*row9*', TIMERANGE => [0, 1568203111265]}

  NOTE

  TimeRange specifies the query time range. The time in the range is a UNIX timestamp, which is the number of milliseconds that have elapsed since the Unix epoch.

– Java API

- Query data in cold storage without the **HOT_ONLY** hint.

```
TableName tableName = TableName.valueOf("chsTable");
Table table = connection.getTable(tableName);
Scan scan = new Scan();
ResultScanner scanner = table.getScanner(scan);
```

- Query data in hot storage with the **HOT_ONLY** hint.

```
Scan scan = new Scan();
scan.setAttribute(HBaseConstants.HOT_ONLY, Bytes.toBytes(true));
```

- Query data within a time range that is specified by **TimeRange**. The system determines whether the query hits cold or hot data based on the values of **TimeRange** and **COLD_BOUNDARY**.

```
Scan scan = new Scan();
scan.setTimeRange(0, 1568203111265);
```

NOTE

TimeRange specifies the query time range. The time in the range is a UNIX timestamp, which is the number of milliseconds that have elapsed since the Unix epoch.

- Prioritizing hot data query

  HBase can search cold storage and hot storage for SCAN queries, for example, that are submitted to search all records of a customer. The query results are returned based on the timestamps when the data records are written in descending order. In most cases, hot data appears before cold data. If the SCAN queries do not carry the **HOT_ONLY** hint, HBase must scan data in both cold and hot storage. As a result, the query takes more time. If you prioritize hot data query, HBase preferentially queries hot data. Cold data is queried only when the number of rows in hot storage is less than the minimum number of rows to be queried. This reduces access to cold storage and improves the response speed.

  - Shell

    scan '*hot_cold_table*', {STARTROW =>'*row1*', STOPROW=>'*row9*',COLD_HOT_MERGE=>true}

  - Java API

```
TableName tableName = TableName.valueOf("hot_cold_table");
Table table = connection.getTable(tableName);
Scan scan = new Scan();
scan.setAttribute(HBaseConstants.COLD_HOT_MERGE, Bytes.toBytes(true));
scanner = table.getScanner(scan);
```

- Major compaction
  - Shell
    - Merge hot data areas of all partitions in a table.

      major_compact '*hot_cold_table*', nil, 'NORMAL', 'HOT'

    - Merge cold data areas of all partitions in a table.

      major_compact '*hot_cold_table*', nil, 'NORMAL', 'COLD'

    - Merge hot and cold data areas of all partitions in a table.

      major_compact '*hot_cold_table*', nil, 'NORMAL', 'ALL'

  - Java API
    - Merge hot data areas of all partitions in a table.

```
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("hot_cold_table");
admin. majorCompact (tableName,null, CompactType.NORMAL,
CompactionScopeType.HOT);
```


- Merge cold data areas of all partitions in a table.

```
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("hot_cold_table");
admin. majorCompact (tableName,null, CompactType.NORMAL,
CompactionScopeType.COLD);
```

- Merge hot and cold data areas of all partitions in a table.

```
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("hot_cold_table");
admin. majorCompact (tableName,null, CompactType.NORMAL,
CompactionScopeType.ALL);
```

9.6.6 Configuring RSGroup to Manage RegionServer Resource

Scenario

A large number of HBase data nodes need to be allocated to specific services based on the service scale so that resources can be exclusively used. When you allocate RegionServes to a RSGroup with the DR feature enabled for AZs, ensure that each AZ must have a RegionServer of the RSGroup so that the DR feature can take effect to ensure service reliability.

Prerequisites

- You have logged in to Manager.
- The login role has the Manager administrator rights.
- The minimum number of nodes in the RSGroup is set to the maximum value calculated in the following three scenarios.
  - To ensure service reliability, a certain number of redundant RegionServer nodes must be configured in the RSGroup to ensure that the number of redundant nodes is greater than the value calculated based on the following formula: (Total number of regions of service tables in RSGroup/ 2000) x 50%.
  - If the system catalog is in an independent RSGroup, ensure that the number of nodes in the RSGroup is greater than 2.
  - To ensure that rolling restart is not affected, if the total number of RegionServer nodes is less than 300, the number of nodes in a single RSGroup must be greater than or equal to 3. If the total number of RegionServer nodes is greater than or equal to 300, the number of nodes in a single RSGroup must be greater than or equal to the value calculated based on the following formula: Number of nodes x 1% + 1.

Possible Impact

- An RSGroup limits the available RegionServer nodes that a Region can be transferred to. If some nodes in the RSGroup are faulty or restarted in rolling mode, an alarm indicating that the number of Regions exceeds the threshold may be triggered, and the service performance may deteriorate.
- If a large number of region transfer tasks are generated when the RSGroup modification request is submitted, related RSGroup operations will fail. You need to observe the Region transfer status on the WebUI page and perform subsequent operations after transfer tasks are complete.

Configuring an RSGroup

Creating an RSGroup

Step 1 On FusionInsight Manager, choose **Cluster** > **Services** > **HBase** > **RSGroup Management**.

Step 2 Click **Add**. On the displayed page, enter the name of the RSGroup to be added. The name can contain 1 to 120 characters, including digits, letters, and underscores (_). Click **OK**.

Viewing an RSGroup

Step 3 Locate the row that contains the target RSGroup and click **View** in the **Operation** column. In the displayed dialog box, view the details about RegionServers and tables in the RSGroup.

NOTE

default is the default RSGroup of HBase. All RegionServer nodes that have been started and are not manually added to other RSGroups will be added to the **default** RSGroup.

Changing the name of an RSGroup

Step 4 Locate the row that contains the target RSGroup and click **Change Name** in the **Operation** column. In the displayed dialog box, enter a new name of the RSGroup and click **OK**. The new name must be unique.

Modifying an RSGroup

Step 5 Click the name of the target RSGroup to go to the **Modify RSGroup** page.

Step 6 Select the RegionServer instance to be allocated and click **Next**.

NOTE

- Only RegionServer instances (one or more) from the same RSGroup can be selected at a time. If the running status of RegionServers in the default RSGroup is not good, the RegionServers cannot be selected for allocation. To allocate RegionServer instances from different RSGroups, perform modification operations multiple times.
- When the cross-AZ feature is enabled, ensure that RegionServer instances of the RSGroup are allocated to each AZ. This constraint does not apply to the RSGroup allocated before the cross-AZ feature is enabled.

Step 7 Select the table to be allocated and click **Next**.

NOTE

- Only tables (one or more) from the same RSGroup can be selected at a time. To allocate RegionServer instances from different RSGroups, perform modification operations multiple times.
- The RegionServers and tables selected for allocation during RSGroup modification must belong to the same RSGroup.
- If only tables are selected during RS group modification and no RegionServer exists in the RSGroup before allocation, the modification fails.

Step 8 Click **Submit**. After the modification is successful, the RSGroup list page is displayed.

When the system displays a message indicating that the task is added to the queue, the RSGroup list page is displayed. The RSGroup modification request has

entered the task queue. Check whether the region transfer is complete on the native page as prompted and ensure that the task is successfully executed before performing subsequent operations.

Deleting an RSGroup

Step 9 On the **RSGroup Management** page, select the RSGroup to be deleted, click **Delete**, and click **OK**.

NOTE

The possible causes and solutions to RSGroup deletion failures are as follows:

- The **default** group cannot be deleted.
- The RSGroup still contains RegionServers or tables. Allocate the RegionServers or tables in the current RSGroup to another RSGroup, and then delete the RSGroup.

----End

9.6.7 Checking Slow and Oversized HBase Requests

Scenario

Query information about slow or oversized requests on the HBase shell CLI. Slow requests refer to the requests whose RPC response exceeds the threshold (value of **hbase.ipc.warn.response.time** set in the HBase service, **3000** ms by default) after you run the **hbase shell** command to query the server. Oversized requests refer to the requests whose data volume returned for each RPC exceeds the threshold (value of **hbase.ipc.warn.response.size** set in the HBase service, **5** MB by default) when you run the **hbase shell** command to query the server.

By default, each RegionServer node caches the latest 256 slow requests and oversized requests. You can adjust the cache size by configuring the **hbase.regionserver.slowlog.ringbuffer.size** parameter on the HBase service page in FusionInsight Manager.

NOTE

This section applies to MRS 3.3.0 or later.

Command Description

This operation involves the following new HBase shell commands:

- **get_slowlog_responses**: queries information about slow requests.
- **get_largelog_responses**: queries information about oversized requests.
- **clear_slowlog_responses**: clears data cached in RegionServers.

You can run the following command in the HBase shell to view how to use related commands:

help '*cmdName*'

For example, run the **help 'clear_slowlog_responses'** command to view the usage of the **clear_slowlog_responses** command.


```
hbase:010:0> help 'clear_slowlog_responses'
Clears SlowLog Responses maintained by each or specific RegionServers.
Specify array of server names for specific RS. A server name is
the host, port plus startcode of a RegionServer.
e.g.: host187.example.com,60020,1289493121758 (find servername in
master ui or when you do detailed status in shell)

Examples:

  hbase> clear_slowlog_responses                                  => clears slowlog responses from all RS
  hbase> clear_slowlog_responses ['SERVER_NAME1', 'SERVER_NAME2']  => clears slowlog responses from SERVER_NAME1,
                                                                     SERVER_NAME2
```

Viewing Request Information

The usage and supported parameters of **get_slowlog_responses** and **get_largelog_responses** are the same. This part describes how to use **get_slowlog_responses**.

You have logged in to the HBase shell CLI by referring to **Using the HBase Client**.

- Run the following command to view the slow requests of all RegionServers:

  get_slowlog_responses '*', {'LIMIT' => *50*}

  The **LIMIT** parameter specifies the number of records returned by each RegionServer. If this parameter is not specified, 10 records are returned by default.

- Run the following command to view the slow requests of a specified RegionServer:

  get_slowlog_responses ['*SERVER_NAME1***', '***SERVER_NAME2***']**

  SERVER_NAME1 and *SERVER_NAME2* indicate the server names of the RegionServers whose slow requests are to be viewed. To view the server names, log in to FusionInsight Manager, choose **Cluster** > **Services** > **HBase**, click the hyperlink next to **HMaster WebUI** to access the HBase web UI, obtain the server names of all RegionServers from **ServerName** on the **Base Stats** tab in the **Region Servers** area. One or more parameters are supported.

- Run the following command to view the slow requests of a specified table and region:

  get_slowlog_responses '*', {'TABLE_NAME' => '*t1***'}**

  get_slowlog_responses '*', {'REGION_NAME' => '*hbase:meta,,1***'}**

  get_slowlog_responses '*', {'REGION_NAME' => '*hbase:meta,,1***', 'TABLE_NAME' => '***t1***', 'FILTER_BY_OP' => 'AND'}**

  TABLE_NAME and **REGION_NAME** indicate the names of the specified table and region, respectively. If **'FILTER_BY_OP' => 'AND'** is specified, each returned result must match all specified conditions. If the parameter is not specified, only any condition needs to be matched.

- Run the following command to view the slow requests of a specified user and client:

  get_slowlog_responses '*', {'USER' => '*user_name***', 'CLIENT_IP' => '***192.162.1.40:60225***'}**

  USER and **CLIENT_IP** indicate the username, client IP address, and port number to be matched. If **'FILTER_BY_OP' => 'AND'** is specified, the result that matches both **USER** and **CLIENT_IP** is returned. If the parameter is not specified, only **USER** or **CLIENT_IP** needs to be matched.

Clearing Request Information

You have logged in to the HBase shell CLI by referring to **Using the HBase Client**.

Run the following command to clear the cache of slow requests and oversized requests of all RegionServers or a specified RegionServer:

clear_slowlog_responses

clear_slowlog_responses ['*SERVER_NAME1***', '***SERVER_NAME2***']**

9.6.8 Configuring HBase Table-Level Overload Control

Scenario

When the HBase requests soar in a short period of time, the system is overloaded. As a result, the P99 latency of requests increases, which severely affects services that relay on quick responses. HBase table-level overload prevention is used to control request latency of core tables (core services).

NOTE

This section applies only to MRS 3.3.1 and later.

Procedure

Step 1 Modify the properties of core tables and set table-level priorities.

1. Log in to the node where the HBase client is installed as the client installation user and configure environment variables.

   cd *HBase client installation directory*

   source bigdata_env

2. If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to authenticate the user:

   kinit *Component service user*

3. Run the following commands to log in to the HBase client and modify the table description:

   hbase shell

   alter '*test_table***', PRIORITY=>'1'**

   NOTE

   - The table priority can be set through the **PRIORITY** property. When the value of **PRIORITY** is greater than or equal to 1, the priority is high. You are advised to set **PRIORITY** to **1**.
   - You can specify the **PRIORITY** property when creating a core table. For example:

     create '*test_table***','cf',PRIORITY=>'1'**

Step 2 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HBase** > **Configurations** > **All Configurations**, search for the parameters listed in **Table 9-11**, and change the parameter values.

Table 9-11 Parameters for HBase table-level overload control

Parameter	Description	Suggestion
hbase.ipc.server.default.callqueue.size.overload.threshold	RegionServer queue threshold. When the percentage of the request size in the queue exceeds the threshold, requests to low-priority tables are discarded. This configuration is used to control the request latency of core tables. When regular request restriction is enabled for protecting internal high-priority requests (the value of **hbase.ipc.server.max.default.callqueue.size.ratio** is not 0), the queue overload threshold is limited by the maximum percentage of regular requests in the queue, the final effective value is the product of the two values.	• The number of core table requests and latency requirements must be considered. A smaller value is recommended. Generally, the value ranges from 0.5 to 1.0. • The two types of overload control can be enabled independently or at the same time. RegionServer queue overload control is used when there is a large number of requests. RegionServer handler overload control is used when there is a large number of concurrent requests.
hbase.ipc.server.handler.overload.threshold	RegionServer handler threshold. When the percentage of active handlers exceeds the threshold, requests to low-priority tables are discarded. This configuration is used to control the request latency of core tables.	

Step 3 Click **Save**. In the displayed dialog box, click **OK**.

Step 4 Click **Instances**, select all RegionServer instances, and choose **More** > **Restart Instance**.

----End

9.6.9 Enabling the HBase Multicast Function

The HBase Multicast function involves two roles: Publisher (HMaster) and Listener (client). You can set **hbase.status.published** to **true** to enable this function. In MRS 3.5.0 and later versions, this function is enabled on the Publisher by default.

When the Multicast function is enabled, the Publisher observes the survival status of the current RegionServer. If there is a Dead RegionServer, the Publisher broadcasts the Dead RegionServer information. The listener configured with the broadcast address receives the information about the Dead RegionServer and automatically deletes the region location cache information on the Dead RegionServer from the connection established by the client. When the region is accessed next time, the listener obtains the latest location information so that the service side can quickly identify the faulty RegionServer and update the region location information cache.

Enabling the HBase Multicast Function

Currently, this function can be enabled on the HBase client running in an IPv4 Linux/Unix environment only. The client must be able to communicate with the service IP address of the HMaster node. Perform the following operations to enable the Multicast function based on service requirements:

Step 1 (Optional) This function is enabled and configured on the HMaster server by default. If there are other NICs besides the NICs mounted to the management IP address and service IP address on the node, log in to FusionInsight Manager, choose **Cluster** > **Services** > **HBase** > **Configurations** > **All Configurations** > **HMaster (Role)** > **Customization**, and add configurations in the **hbase.hmaster.config.expandor** customized parameter. The value of **hbase.status.multicast.ni.name** is the name of the NIC to which the service IP address is mounted.

Step 2 Log in to the node where the HBase client is installed as the client installation user.

Step 3 In *HBase client installation directory*/**HBase/hbase/conf/hbase-site.xml**, add and configure the parameters in **Table 9-12** to enable the Multicast function of the client.

Table 9-12 Parameters for enabling the function

Parameter	Description	Value
hbase.status.published	If this parameter is set to **true**, the Multicast function is enabled on the client.	true
hbase.status.multicast.address.ip	Broadcast address used by the Multicast feature. In an IPv4 network, the broadcast address is **226.1.1.3**.	226.1.1.3
hbase.status.multicast.bind.address.ip	Broadcast address bound to the client listener. In a Linux/Unix environment, the value must be the same as the broadcast address.	226.1.1.3

Parameter	Description	Value
hbase.status.multicast.ni.name	Name of the NIC used by the service IP address if a node has multiple NICs. This configuration is optional for a single-NIC node.	-

Refer to the following configurations:

```
<property>
<name>hbase.status.published</name>
<value>true</value>
</property>
<property>
<name>hbase.status.multicast.address.ip</name>
<value>226.1.1.3</value>
</property>
<property>
<name>hbase.status.multicast.bind.address.ip</name>
<value>226.1.1.3</value>
</property>
```

Step 4 Verify that the Multicast function is enabled on the client.

Start read and write tasks on the HBase client and locate the RegionServer where the target region is located. Restart the RegionServer during read/write task execution. If "ClusterStatusListener: There is a new dead server" is displayed on the client, the multicast function is enabled.

The following describes how to check whether the Multicast function is enabled for read and write tasks executed on the HBase Shell client:

1. Switch to the client installation directory, configure environment variables, and authenticate the user.

   cd *HBase client installation directory*

   source bigdata_env

   kinit *Component service user* (skip this step if Kerberos authentication is disabled for the cluster (the cluster is in normal mode))

2. Log in to the HBase client, create a table, and write data. Do not close the HBase client.

   hbase shell

   create 'test_multicast', 'f1'

   put 'test_multicast','row1','f1:q','value1'

   put 'test_multicast','row2','f1:q','value2'

   put 'test_multicast','row3','f1:q','value3'

3. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HBase**, and click the **HMaster(***Host name***, active)** hyperlink on the right of **HMaster WebUI** to go to the HBase web UI.
4. In the **Tables** area, click the table name **test_multicast**. In the **Table Regions** area, locate the RegionServer where the table region is located.

5. On FusionInsight Manager, choose **Cluster** > **Services** > **HBase** > **Instances**, select the RegionServerServer instance queried in **Step 4.4**, and choose **More** > **Restart Instance** or **Instance Rolling Restart**.
6. If the key log "client.ClusterStatusListener: There is a new dead server" is displayed in the HBase Shell, the multicast function is enabled.

----End

9.7 HBase Performance Tuning

9.7.1 Improving the Batch Loading Efficiency of HBase BulkLoad

Scenario

HBase BulkLoad uses MapReduce jobs to directly generate files that comply with the internal data format of HBase, and then loads the generated StoreFiles to a running cluster. Compared with HBase APIs, BulkLoad saves more CPU and network resources.

ImportTSV is an HBase table data loading tool.

Prerequisites

When using BulkLoad, the output path of the file has been specified using the **Dimporttsv.bulk.output** parameter.

Procedure

Add the **Table 9-13** parameters to the BulkLoad command when performing a batch loading task.

Table 9-13 Parameter for improving BulkLoad efficiency

Parameter	Description	Value
-Dimporttsv.mapper.class	The construction of key-value pairs is moved from the user-defined mapper to reducer to improve performance. The mapper only needs to send the original text in each row to the reducer. The reducer parses the record in each row and creates a key-value) pair. **NOTE** When this parameter is set to **org.apache.hadoop.hbase.mapreduce.TsvImporterByteMapper**, this parameter is used only when the batch loading command without the *HBASE_CELL_VISIBILITY OR HBASE_CELL_TTL* option is executed. The **org.apache.hadoop.hbase.mapreduce.TsvImporterByteMapper** provides better performance.	org.apache.hadoop.hbase.mapreduce.TsvImporterByteMapper and org.apache.hadoop.hbase.mapreduce.TsvImporterTextMapper

9.7.2 Improving HBase Continuous Put Performance

Scenario

In the scenario where a large number of requests are continuously Put, setting the following two parameters to **false** can greatly improve the Put performance.

- **hbase.regionserver.wal.durable.sync**
- **hbase.regionserver.hfile.durable.sync**

When the performance is improved, there is a low probability that data is lost if three DataNodes are faulty at the same time. Exercise caution when configuring the parameters in scenarios that have high requirements on data reliability.

Procedure

Navigation path for setting parameters:

On FusionInsight Manager, choose **Cluster** > **Services** > **HBase** > **Configurations** > **All Configurations**. Enter the parameter name in the search box, and change the value.

Table 9-14 Parameters for improving put performance

Parameter	Description	Value
hbase.wal.hsync	Specifies whether to enable WAL file durability to make the WAL data persistence on disks. If this parameter is set to **true**, the performance is affected because each WAL file is synchronized to the disk by the Hadoop fsync.	false
hbase.hfile.hsync	Specifies whether to enable the HFile durability to make data persistence on disks. If this parameter is set to **true**, the performance is affected because each Hfile file is synchronized to the disk by the Hadoop fsync.	false

9.7.3 Improving HBase Put and Scan Performance

Scenario

HBase has many configuration parameters related to read and write performance. The configuration parameters need to be adjusted based on the read/write request loads. This section describes how to optimize read and write performance by modifying the RegionServer configurations.

Procedure

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HBase** > **Configurations**, and configure the following parameters to improve the HBase data read and write performance.

- JVM GC parameters

  Suggestions on setting the RegionServer **GC_OPTS** parameter:

  - Set **-Xms** and **-Xmx** to the same value based on your needs. Increasing the memory can improve the read and write performance. For details, see the description of **hfile.block.cache.size** in **Table 9-16** and **hbase.regionserver.global.memstore.size** in **Table 9-15**.
  - Set **-XX:NewSize** and **-XX:MaxNewSize** to the same value. You are advised to set the value to **512M** in low-load scenarios and **2048M** in high-load scenarios.
  - Set **X-XX:CMSInitiatingOccupancyFraction** to be less than and equal to 90, and it is calculated as follows: **100 x (hfile.block.cache.size + hbase.regionserver.global.memstore.size + 0.05)**.

  - **-XX:MaxDirectMemorySize** indicates the non-heap memory used by the JVM. You are advised to set this parameter to **512M** in low-load scenarios and **2048M** in high-load scenarios.

    NOTE

    The **-XX:MaxDirectMemorySize** parameter is not used by default. If you need to set this parameter, add it to the **GC_OPTS** parameter.

- Put parameters

  RegionServer processes the data of the Put request and writes the data to MemStore and HLog.

  - When the size of MemStore reaches the value of **hbase.hregion.memstore.flush.size**, MemStore is updated to HDFS to generate HFiles.
  - Compaction is triggered when the number of HFiles in the column cluster of the current Region reaches the value of **hbase.hstore.compaction.min**.
  - If the number of HFiles in the column cluster of the current region reaches the value of **hbase.hstore.blockingStoreFiles**, the operation of refreshing the MemStore and generating HFiles is blocked. As a result, the Put request is blocked.

  Table 9-15 Put parameters

Parameter	Description	Default Value
hbase.wal.hsync	Indicates whether each WAL is persistent to disks. For details, see **Improving HBase Continuous Put Performance**.	true
hbase.hfile.hsync	Indicates whether HFile write operations are persistent to disks. For details, see **Improving HBase Continuous Put Performance**.	true

Parameter	Description	Default Value
hbase.hregion.memstore.flush.size	If the size of MemStore (unit: Byte) exceeds a specified value, MemStore is flushed to the corresponding disk. The value of this parameter is checked by each thread running **hbase.server.thread.wakefrequency**. It is recommended that you set this parameter to an integer multiple of the HDFS block size. You can increase the value if the memory is sufficient and the put load is heavy.	134217728
hbase.regionserver.global.memstore.size	Updates the size of all MemStores supported by the RegionServer before locking or forcible flush. It is recommended that you set this parameter to **hbase.hregion.memstore.flush.size x Number of regions with active writes/ RegionServer GC -Xmx**. The default value is **0.4**, indicating that 40% of RegionServer GC -Xmx is used.	0.4
hbase.hstore.flusher.count	Indicates the number of MemStore flush threads. You can increase the parameter value in heavy-put-load scenarios.	2
hbase.regionserver.thread.compaction.small	Indicates the number of small compaction threads. You can increase the parameter value in heavy-put-load scenarios.	10

Parameter	Description	Default Value
hbase.hstore.blockingStoreFiles	If the number of HStoreFile files in a Store exceeds the specified value, the update of the HRegion will be locked until a compression is completed or the value of **base.hstore.blockingWaitTime** is exceeded. Each time MemStore is flushed, a StoreFile file is written into MemStore. Set this parameter to a larger value in heavy-put-load scenarios.	15

- Scan parameters

  Table 9-16 Scan parameters

Parameter	Description	Default Value
hbase.client.scanner.timeout.period	Client and RegionServer parameters, indicating the lease timeout period of the client executing the scan operation. You are advised to set this parameter to an integer multiple of 60000 ms. You can set this parameter to a larger value when the read load is heavy. The unit is milliseconds.	60000
hfile.block.cache.size	Indicates the data cache percentage in the RegionServer GC –Xmx. You can increase the parameter value in heavy-read-load scenarios, in order to improve cache hit ratio and performance. It indicates the percentage of the maximum heap (-Xmx setting) allocated to the block cache of HFiles or StoreFiles.	When offheap is disabled, the default value is **0.25**. When offheap is enabled, the default value is **0.1**.

- Handler parameters

Table 9-17 Handler parameters

Parameter	Description	Default Value
hbase.regionserver.handler.count	Indicates the number of RPC server instances on RegionServer. The recommended value ranges from 200 to 400.	200
hbase.regionserver.metahandler.count	Indicates the number of program instances for processing prioritized requests. The recommended value ranges from 200 to 400.	200

9.7.4 Improving HBase Real-Time Write Efficiency

Scenario

Scenarios where data needs to be written to HBase in real time, or large-scale and consecutive put scenarios

Prerequisites

The HBase put or delete interface can be used to save data to HBase.

Procedure

- **Data writing server tuning**

  Parameter portal: Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Configurations** > **All Configurations**. The All Configurations page is displayed, modify related parameter values.

  Table 9-18 Configuration items that affect real-time data writing

Parameter	Description	Default Value
hbase.wal.hsync	Controls the synchronization degree when HLogs are written to the HDFS. If the value is **true**, HDFS returns only when data is written to the disk. If the value is **false**, HDFS returns when data is written to the OS cache. Set the parameter to **false** to improve write performance.	true

Parameter	Description	Default Value
hbase.hfile.hsync	Controls the synchronization degree when HFiles are written to the HDFS. If the value is **true**, HDFS returns only when data is written to the disk. If the value is **false**, HDFS returns when data is written to the OS cache. Set the parameter to **false** to improve write performance.	true

Parameter	Description	Default Value
GC_OPTS	You can increase HBase memory to improve HBase performance because read and write operations are performed in HBase memory. **HeapSize** and **NewSize** need to be adjusted. When you adjust **HeapSize**, set **Xms** and **Xmx** to the same value to avoid performance problems when JVM dynamically adjusts **HeapSize**. Set **NewSize** to 1/8 of **HeapSize**. • **HMaster**: If HBase clusters enlarge and the number of Regions grows, properly increase the **GC_OPTS** parameter value of the HMaster. • **RegionServer**: A RegionServer needs more memory than an HMaster. If sufficient memory is available, increase the **HeapSize** value. **NOTE** When the value of **HeapSize** for the active HMaster is 4 GB, the HBase cluster can support 100,000 regions. Empirically, each time 35,000 regions are added to the cluster, the value of **HeapSize** must be increased by 2 GB. It is recommended that the value of **HeapSize** for the active HMaster not exceed 32 GB.	• HMaster -server -Xms4G -Xmx4G -XX:NewSize=512M -XX:MaxNewSize=512M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M

Parameter	Description	Default Value
		• Region Server -server -Xms6G -Xmx6G -XX:NewSize=1024M -XX:MaxNewSize=1024M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M

Parameter	Description	Default Value
hbase.regionserver.handler.count	Indicates the number of RPC server instances started on RegionServer. If the parameter is set to an excessively large value, threads will compete fiercely. If the parameter is set to an excessively small value, requests will be waiting for a long time in RegionServer, reducing the processing capability. You can add threads based on resources. It is recommended that the value be set to **100** to **300** based on the CPU usage.	200
hbase.hregion.max.filesize	Indicates the maximum size of an HStoreFile, in bytes. If the size of any HStoreFile exceeds the value of this parameter, the managed Hregion is divided into two parts.	10737418240
hbase.hregion.memstore.flush.size	On the RegionServer, when the size of MemStore that exists in memory of write operations exceeds **memstore.flush.size**, MemStoreFlusher performs the Flush operation to write the MemStore to the corresponding store in the format of HFile. If RegionServer memory is sufficient and active Regions are few, increase the parameter value and reduce compaction times to improve system performance. The Flush operation may be delayed after it takes place. Write operations continue and MemStore keeps increasing during the delay. The maximum size of MemStore is: **memstore.flush.size** x **hbase.hregion.memstore.block.multiplier**. When the MemStore size exceeds the maximum value, write operations are blocked. Properly increasing the value of **hbase.hregion.memstore.block.multiplier** can reduce the blocks and make performance become more stable. Unit: byte	134217728

Parameter	Description	Default Value
hbase.regionserver.global.memstore.size	Updates the size of all MemStores supported by the RegionServer before locking or forcible flush. On the RegionServer, the MemStoreFlusher thread performs the flush. The thread regularly checks memory occupied by write operations. When the total memory volume occupied by write operations exceeds the threshold, MemStoreFlusher performs the flush. Larger MemStore will be flushed first and then smaller ones until the occupied memory is less than the threshold. Threshold = hbase.regionserver.global.memstore.size x hbase.regionserver.global.memstore.size.lower.limit x HBase_HEAPSIZE **NOTE** The sum of the parameter value and the value of **hfile.block.cache.size** cannot exceed 0.8, that is, memory occupied by read and write operations cannot exceed 80% of **HeapSize**, ensuring stable running of other operations.	0.4

Parameter	Description	Default Value
hbase.hstore.blockingStoreFiles	Check whether the number of files is larger than the value of **hbase.hstore.blockingStoreFiles** before you flush regions. If it is larger than the value of **hbase.hstore.blockingStoreFiles**, perform a compaction and configure **hbase.hstore.blockingWaitTime** to 90s to make the flush delay for 90s. During the delay, write operations continue and the MemStore size keeps increasing and exceeds the threshold (**memstore.flush.size** x **hbase.hregion.memstore.block.multiplier**), blocking write operations. After compaction is complete, a large number of writes may be generated. As a result, the performance fluctuates sharply. Increase the value of **hbase.hstore.blockingStoreFiles** to reduce block possibilities.	15
hbase.regionserver.thread.compaction.throttle	The compression whose size is greater than the value of this parameter is executed by the large thread pool. The unit is bytes. Indicates a threshold of a total file size for compaction during a Minor Compaction. The total file size affects execution duration of a compaction. If the total file size is large, other compactions or flushes may be blocked.	1610612736
hbase.hstore.compaction.min	Indicates the minimum number of HStoreFiles on which minor compaction is performed each time. When the size of a file in a Store exceeds the value of this parameter, the file is compacted. You can increase the value of this parameter to reduce the number of times that the file is compacted. If there are too many files in the Store, read performance will be affected.	6

Parameter	Description	Default Value
hbase.hstore.compaction.max	Indicates the maximum number of HStoreFiles on which minor compaction is performed each time. The functions of the parameter and **hbase.hstore.compaction.max.size** are similar. Both are used to limit the execution duration of one compaction.	10
hbase.hstore.compaction.max.size	If the size of an HFile is larger than the parameter value, the HFile will not be compacted in a Minor Compaction but can be compacted in a Major Compaction. The parameter is used to prevent HFiles of large sizes from being compacted. After a Major Compaction is forbidden, multiple HFiles can exist in a Store and will not be merged into one HFile, without affecting data access performance. The unit is byte.	9223372036854775807
hbase.hregion.majorcompaction	Main compression interval of all HStoreFile files in a region. The unit is milliseconds. Execution of Major Compactions consumes much system resources and will affect system performance during peak hours. If service updates, deletion, and reclamation of expired data space are infrequent, set the parameter to **0** to disable Major Compactions. If you must perform a Major Compaction to reclaim more space, increase the parameter value and configure the **hbase.offpeak.end.hour** and **hbase.offpeak.start.hour** parameters to make the Major Compaction be triggered in off-peak hours.	604800000

Parameter	Description	Default Value
• hbase.regionserver.maxlogs • hbase.regionserver.hlog.blocksize	• Indicates the threshold for the number of HLog files that are not flushed on a RegionServer. If the number of HLog files is greater than the threshold, the RegionServer forcibly performs flush operations. • Indicates the maximum size of an HLog file. If the size of an HLog file is greater than the value of this parameter, a new HLog file is generated. The old HLog file is disabled and archived. The two parameters determine the number of HLogs that are not flushed in a RegionServer. When the data volume is less than the total size of MemStore, the flush operation is forcibly triggered due to excessive HLog files. In this case, you can adjust the values of the two parameters to avoid forcible flush. Unit: byte	• 32 • 134217728

- **Data writing client tuning**

  It is recommended that data is written in Put List mode if necessary, which greatly improves write performance. The length of each put list needs to be set based on the single put size and parameters of the actual environment. You are advised to do some basic tests before configuring parameters.

- **Data table writing design optimization**

  Set the following table parameters in the hbase shell to improve the data write performance of HBase.

Table 9-19 Parameters affecting real-time data writing

Parameter	Description	Default Value
COMPRESSION	The compression algorithm compresses blocks in HFiles. For compressible data, configure the compression algorithm to efficiently reduce disk I/Os and improve performance. **NOTE** Some data cannot be efficiently compressed. For example, a compressed figure can hardly be compressed again. The common compression algorithm is SNAPPY, because it has a high encoding/decoding speed and acceptable compression rate.	NONE
BLOCKSIZE	Different block sizes affect HBase data read and write performance. You can configure sizes for blocks in an HFile. Larger blocks have a higher compression rate. However, they have poor performance in random data read, because HBase reads data in a unit of blocks. Set the parameter to 128 KB or 256 KB to improve data write efficiency without greatly affecting random read performance. The unit is byte.	65536
IN_MEMORY	Whether to cache table data in the memory first, which improves data read performance. If you will frequently access some small tables, set the parameter.	false

9.7.5 Improving HBase Real-Time Read Efficiency

Scenario

Scenarios where the performance of reading HBase data needs to be improved.

Prerequisites

The get or scan interface of HBase has been invoked and data is read in real time from HBase.

Procedure

- **Data reading server tuning**

  Parameter portal: Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Configurations** > **All Configurations**. The All Configurations page is displayed, modify related parameter values to improve the HBase data read performance.

Table 9-20 Configuration items that affect real-time data reading

Parameter	Description	Default Value
GC_OPTS	You can increase HBase memory to improve HBase performance because read and write operations are performed in HBase memory. **HeapSize** and **NewSize** need to be adjusted. When you adjust **HeapSize**, set **Xms** and **Xmx** to the same value to avoid performance problems when JVM dynamically adjusts **HeapSize**. Set **NewSize** to 1/8 of **HeapSize**. ● **HMaster**: If HBase clusters enlarge and the number of Regions grows, properly increase the **GC_OPTS** parameter value of the HMaster. ● **RegionServer**: A RegionServer needs more memory than an HMaster. If sufficient memory is available, increase the **HeapSize** value. **NOTE** When the value of **HeapSize** for the active HMaster is 4 GB, the HBase cluster can support 100,000 Regions. Empirically, each time 35,000 Regions are added to the cluster, the value of **HeapSize** must be increased by 2 GB. It is recommended that the value of **HeapSize** for the active HMaster not exceed 32 GB.	● HMaster -server -Xms4G -Xmx4G -XX:NewSize=512M -XX:MaxNewSize=512M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -

Parameter	Description	Default Value
		XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M • Region Server -server -Xms6G -Xmx6G -XX:NewSize=1024M -XX:MaxNewSize=1024M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGC

Parameter	Description	Default Value
		DateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M
hbase.regionserver.handler.count	Indicates the number of requests that RegionServer can process concurrently. If the parameter is set to an excessively large value, threads will compete fiercely. If the parameter is set to an excessively small value, requests will be waiting for a long time in RegionServer, reducing the processing capability. You can add threads based on resources. It is recommended that the value be set to 100 to 300 based on the CPU usage.	200
hfile.block.cache.size	HBase cache sizes affect query efficiency. Set cache sizes based on query modes and query record distribution. If random query is used to reduce the hit ratio of the buffer, you can reduce the buffer size.	When **offheap** is disabled, the default value is **0.25**. When **offheap** is enabled, the default value is **0.1**.

NOTE

If read and write operations are performed at the same time, the performance of the two operations affects each other. If flush and compaction operations are frequently performed due to data writes, a large number of disk I/O operations are occupied, affecting read performance. If a large number of compaction operations are blocked due to write operations, multiple HFiles exist in the region, affecting read performance. Therefore, if the read performance is unsatisfactory, you need to check whether the write configurations are proper.

- **Data reading client tuning**

  When scanning data, you need to set **caching** (the number of records read from the server at a time. The default value is **1**.). If the default value is used, the read performance will be extremely low.

  If you do not need to read all columns of a piece of data, specify the columns to be read to reduce network I/O.

If you only need to read the row key, add a filter (FirstKeyOnlyFilter or KeyOnlyFilter) that only reads the row key.

- **Data table reading design optimization**

  Set the following table parameters in the HBase shell to improve the performance of reading HBase data in real time.

  Table 9-21 Parameters affecting real-time data reading

Parameter	Description	Default Value
COMPRESSION	The compression algorithm compresses blocks in HFiles. For compressible data, configure the compression algorithm to efficiently reduce disk I/Os and improve performance. **NOTE** Some data cannot be efficiently compressed. For example, a compressed figure can hardly be compressed again. The common compression algorithm is SNAPPY, because it has a high encoding/decoding speed and acceptable compression rate.	NONE
BLOCKSIZE	Different block sizes affect HBase data read and write performance. You can configure sizes for blocks in an HFile. Larger blocks have a higher compression rate. However, they have poor performance in random data read, because HBase reads data in a unit of blocks. Set the parameter to 128 KB or 256 KB to improve data write efficiency without greatly affecting random read performance. The unit is byte.	65536
DATA_BLOCK_ENCODING	Encoding method of the block in an HFile. If a row contains multiple columns, set **FAST_DIFF** to save data storage space and improve performance.	NONE

9.7.6 Accelerating HBase Compaction During Off-Peak Hours

Scenario

HBase allows you to set compaction throughput during off-peak hours. A large throughput can speed up compaction execution and reduce impacts on services during peak hours.

(This operation is supported only for MRS 3.3.0 or later.)

Configuring HBase Compaction Throughput Parameters

Log in to FusionInsight Manager, choose **Cluster** > **Service** > **HBase**, and click **Configuration**. In the search box, search for the parameters listed in **Table 9-22** and change the parameter values as you need. The parameter settings take effect dynamically. After the modification is saved, log in to the **hbase shell** CLI and run the **update_all_config** command to update the configuration. You do not need to restart the instance.

NOTE

To enable the HBase compaction throughput settings, neither **hbase.offpeak.start.hour** nor **hbase.offpeak.end.hour** must be **-1**.

Table 9-22 HBase compaction throughput parameters

Parameter	Description	Default Value
hbase.offpeak.start.hour	Start time of off-peak hours of the HBase cluster. The value must be an integer from **-1** to **23**. If the value is **-1**, HBase compaction throughput will not be used.	-1
hbase.offpeak.end.hour	End time of off-peak hours of the HBase cluster. The value must be an integer from **-1** to **23**. If the value is **-1**, HBase compaction throughput will not be used.	-1
hbase.hstore.compaction.throughput.offpeak	Compaction throughput during off-peak hours. The unit is byte/s.	104857600

Configuration Example

Step 1 Log in to FusionInsight Manager, choose **Cluster** > **Service** > **HBase**, and click **Configuration**. Set off-peak hours as needed. The following are examples:

- If **hbase.offpeak.start.hour** is **18** and **hbase.offpeak.end.hour** is **23**, off-peak hours are 18:00 to 23:00 every day.
- If **hbase.offpeak.start.hour** is **23** and **hbase.offpeak.end.hour** is **8**, off-peak hours are 23:00 to 8:00 the next day.

Step 2 During off-peak hours, log in to FusionInsight Manager, choose **Cluster** > **Service** > **HBase**, and click **Chart**. Check whether the value of **Compaction Queue Size-All Instances** keeps increasing and whether the values of some RegionServers in **Traffic of the RegionServer Compaction Operations-All Instances** have reached or exceeded the value of **hbase.hstore.compaction.throughput.offpeak**.

- If they are, set **hbase.hstore.compaction.throughput.offpeak** to a larger value based on the cluster disk usage and go to **Step 3**.
- If they are not, no further action is required.

Step 3 Check whether the value of **P99 Percentile RegionServer RPC Request Response Time-All Instances** keeps increasing in the HBase chart.

- If it does, go to **Step 4**.
- If it does not, no further action is required.

Step 4 Check whether the value of **Disk IO Utilization** of the host where the RegionServer is located exceeds 90%.

- If it does, reduce the write speed or expand the disk capacity.
- If it does not, no further action is required.

----End

9.7.7 Tuning HBase JVM Parameters

Scenario

When the number of clusters reaches a certain scale, the default settings of the Java virtual machine (JVM) cannot meet the cluster requirements. In this case, the cluster performance deteriorates or the clusters may be unavailable. Therefore, JVM parameters must be properly configured based on actual service conditions to improve the cluster performance.

Procedure

Navigation path for setting parameters:

The JVM parameters related to the HBase role must be configured in the **hbase-env.sh** file in the **${BIGDATA_HOME}/FusionInsight_HD_*/install/FusionInsight-HBase-2.2.3/hbase/conf/** directory of the node where the HBase service is installed.

Each role has JVM parameter configuration variables, as shown in **Table 9-23**.

Table 9-23 HBase-related JVM parameter configuration variables

Variable	Affected Role
HBASE_OPTS	All roles of HBase
SERVER_GC_OPTS	All roles on the HBase server, such as Master and RegionServer
CLIENT_GC_OPTS	Client process of HBase
HBASE_MASTER_OPTS	Master of HBase
HBASE_REGIONSERVER_OPTS	RegionServer of HBase
HBASE_THRIFT_OPTS	Thrift of HBase

Configuration example:

```
export HADOOP_NAMENODE_OPTS="-Dhadoop.security.logger=${HADOOP_SECURITY_LOGGER:-INFO,RFAS} -Dhdfs.audit.logger=${HDFS_AUDIT_LOGGER:-INFO,NullAppender} $HADOOP_NAMENODE_OPTS"
```


9.7.8 Optimization for HBase Overload

Scenario

When the HBase service peaks suddenly and a large number of requests are sent to a RegionServer/HMaster in a short period of time, the RegionServer/HMaster is overloaded. If the HBase service is overloaded, the read and write performance of the application deteriorates, GC occurs frequently on the HBase service, and even the service instance restarts.

Currently, HBase can prevent overloading. It can reject oversized requests, protect internal requests, and record improper requests, reducing the impact on HBase services in overload scenarios and ensuring service stability.

This topic is available for MRS 3.3.0 and later versions only.

Sharp Traffic Increase

When service traffic peaks, for example, the number of requests increases by 10 times, you can perform the following operations to manage the traffic:

Step 1 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HBase** > **Chart**, select **Handler** in the chart category on the left, and check whether "Number of Active RegionServer Handlers for Processing User Table Requests-All Instances" is used up for a long time. If they are used up, click **Configure**. The following table lists the RegionServer parameters to be configured.

Table 9-24 Optimizing parameters when RegionServer handlers are used up

Parameter	Description	Optimization
hbase.regionserver.handler.count	Number of RPC server instances started on RegionServer	Increase the value of this parameter. However, the value should be less than or equal to **1000**.
hbase.ipc.server.max.default.callqueue.size.ratio	Maximum percentage of common requests in the RegionServer queue. When the total size of common requests in the queue exceeds the threshold, the requests are discarded.	Adjust the value to about **0.8** to limit the proportion of queues occupied by external requests and protect internal requests.

Step 2 Check whether "*XXX* is too large for table *XXX*" or "Client scan caching *XXX* is too large for table *XXX*" exists in the service run logs on the application side. If yes, improper requests exist. Check the requests and reduce the data volume of each request (reduce the data volume for Put/Delete batch requests and decrease the Caching value for Scan). If services on the service side cannot be optimized temporarily, you can add or modify the following parameters in the ***Client installation directory*/HBase/hbase/conf/hbase-site.xml** file on the application side. (**This only reduces recorded alarm logs but does not relieve overload**.)

Table 9-25 Parameters for reducing recorded alarm logs

Parameter	Description	Optimization
hbase.rpc.rows.warning.threshold	Threshold of the number of data records written, updated, or deleted by the HBase client at a time. If the threshold is exceeded, a log is recorded.	Increase the value of this parameter.
hbase.client.scanner.warning.threshold.scanning.ratio	If the caching of a single scan on the HBase client is too large (40% of the maximum value by default), a log is recorded when the threshold is exceeded.	Change the value of this parameter to **1.0**.

Step 3 If the service side sends too many oversized requests, the server processes the requests slowly. As a result, the requests are stacked and overloaded. **If oversized requests can be considered as abnormal requests**, adjust the parameters in the HBase configuration on FusionInsight Manager to reject the requests. The following table lists the RegionServer parameters to be configured.

Table 9-26 Parameters for rejecting requests

Parameter	Description	Optimization
hbase.ipc.max.request.size	Maximum size of a RegionServer request. If a request is bigger than the specified size, the request is discarded. The default value is **256 MB**.	If the application retried for multiple times and "RPC data length *XXX* of received from *XXX* is greater than max allowed" is displayed in RegionServer logs, reduce the amount of data sent at a time on the application side. If the amount cannot be reduced, you can increase the value of this parameter. It is recommended that the value be less than or equal to **1 GB**.

Parameter	Description	Optimization
hbase.server.keyvalue.maxsize	Maximum size of a single cell for RegionServer write/update operations. If the value of this parameter is exceeded, RegionServer write/update operations are not allowed. The default value is **10 MB**.	If a single cell is too large, the read and write performance is degraded and abnormal data may exist. You can evaluate the data range based on the written data and set the upper limit. If the evaluation cannot be performed, you are advised to retain the default value.
hbase.rpc.rows.size.threshold.reject	Whether to reject a RegionServer request when the number of data operations in the request exceeds the specified limit.	If there is a request contains a large number of write, update, and delete operations on a node, the number of operations may exceed the value of **hbase.rpc.rows.warning.threshold**. In this case, overloading occurs and the performance deteriorates. If this parameter is set to **true**, large requests will be rejected. If the pre-partitioning is improper, too many requests may be rejected. Set this parameter to **true** only when stable.

----End

Server Restart in a Large Number of Regions

When multiple RegionServers of large-scale clusters in a number of regions (more than 100,000) are restarted at the same time, HMaster may be overloaded.

You can configure the parameters listed in **Table 9-27** in the HBase configuration on FusionInsight Manager to accelerate HMaster processing of high-priority requests and reduce HMaster overload.

Table 9-27 Parameters for handling overloading caused by online/offline switches in a large number of regions

Instance Name	Parameter	Description	Optimization
HMaster	hbase.regionserver.metahandler.count	Number of handlers used by HMaster to process high-priority requests	Increase the value of this parameter. However, the value should be less than or equal to **1000**.
	hbase.ipc.server.metacallqueue.read.ratio	Ratio of read queues in a high-priority request queue, which affects the number of meta read/write handlers	Retain the default value **0.5**.
RegionServer	hbase.regionserver.msginterval	Interval for transmitting messages between RegionServer and HMaster	Increase the value of this parameter can release the pressure on HMaster. The recommended value is **15s**.

9.7.9 Enabling CCSMap Functions

Scenario

CompactedConcurrentSkipListMap (CCSMap) optimizes the Memstore data structure and uses less memory in data write scenarios, reducing GC times for higher data write performance. You can enable this feature to handle tasks that require high data write performance.

NOTE

This section applies only to MRS 3.3.1 and later.

Procedure

Step 1 Log in to FusionInsight Manager of the cluster and choose **Cluster** > **Services** > **HBase** > **Configurations** > **All Configurations**.

Step 2 Search for and modify the following parameters to enable the CCSMap feature:

- **hbase.regionserver.memstore.class**: Memstore implementation class. Set this parameter to **org.apache.hadoop.hbase.regionserver.CCSMapMemStore**.
- **hbase.hregion.compacting.memstore.type**: Memstore memory compaction policy. Set this parameter to **NONE**.

Step 3 Click **Save**.

Step 4 Click **Instances**, select all RegionServer instances, choose **More** > **Instance Rolling Restart**, and enter the password of the user to restart the RegionServer instances.

----End

9.7.10 Enabling Succinct Trie

Scenario

Succinct Trie optimizes the HFile Block structure. It uses less cache space, reduces the cache data eviction rate, and improves the cache hit ratio. You can enable this feature to improve performance of tasks that frequently read data.

This section applies only to MRS 3.3.1 and later.

NOTICE

If Succinct Trie is enabled, open-source versions of HFiles are incompatible. If you are using an HFile to migrate data to MRS 3.2.0 or an earlier version, disable this feature and then run the **major compaction** command on the data table to generate a new HFile file.

Procedure

Step 1 Log in to FusionInsight Manager of the cluster and choose **Cluster** > **Services** > **HBase** > **Configurations**.

Step 2 In the search box, search for and modify the configuration in **Table 9-28** to enable Succinct Trie.

Table 9-28 Succinct Trie parameters

Parameter	Description	New Value	Must Be Modified
hbase.write.tries	Whether to enable Succinct Tries • **true**: Enable Succinct Tries • **false**: Disable Succinct Tries	true	Yes

Parameter	Description	New Value	Must Be Modified
hbase.tries.cache.enabled	If this parameter is set to **true**, LoudsTriesLruBlockCache uses off-heap memory to cache index blocks, reducing the eviction rate of index blocks and improving cache efficiency.	true	No
hbase.index.block.cache.size	Cache size ratio of the LoudsTriesLruBlockCache index block to the **blocksize** If the value of **blocksize** is small, you are advised to increase the value.	-	No

Step 3 Click **Save**.

Step 4 Click **Instances**, select all RegionServer instances, choose **More** > **Instance Rolling Restart**, and enter the password of the user to restart the RegionServer instances.

----End

9.8 HBase O&M Management

9.8.1 HBase Log Overview

Log Description

Log path: The default storage path of HBase logs is **/var/log/Bigdata/hbase/***Role name*.

- HMaster: **/var/log/Bigdata/hbase/hm** (run logs) and **/var/log/Bigdata/audit/hbase/hm** (audit logs)
- RegionServer: **/var/log/Bigdata/hbase/rs** (run logs) and **/var/log/Bigdata/audit/hbase/rs** (audit logs)
- ThriftServer: **/var/log/Bigdata/hbase/ts2** (run logs, **ts2** is the instance name) and **/var/log/Bigdata/audit/hbase/ts2** (audit logs, **ts2** is the instance name)

Log archive rule: The automatic log compression and archiving function of HBase is enabled. By default, when the size of a log file exceeds 30 MB, the log file is automatically compressed. The naming rule of a compressed log file is as follows:

<*Original log name*>-<*yyyy-mm-dd_hh-mm-ss*>.[*ID*].**log.zip** A maximum of 20 latest compressed files are reserved. The number of compressed files can be configured on the Manager portal.

Table 9-29 HBase log list

Type	Name	Description
Run logs	hbase-<SSH_USER>-<process_name>-<hostname>.log	HBase system log that records the startup time, startup parameters, and most logs generated when the HBase system is running.
	hbase-<SSH_USER>-<process_name>-<hostname>.out	Log that records the HBase running environment information.
	<process_name>-<SSH_USER>-<DATE>-<PID>-gc.log	Log that records HBase junk collections.
	checkServiceDetail.log	Log that records whether the HBase service starts successfully.
	hbase.log	Log generated when the HBase service health check script and some alarm check scripts are executed.
	sendAlarm.log	Log that records alarms reported after execution of HBase alarm check scripts.
	hbase-haCheck.log	Log that records the active and standby status of HMaster
	stop.log	Log that records the startup and stop processes of HBase.
Audit logs	hbase-audit-<process_name>.log	Log that records HBase security audit.

Log Level

Table 9-30 describes the log levels supported by HBase. The priorities of log levels are FATAL, ERROR, WARN, INFO, and DEBUG in descending order. Logs whose levels are higher than or equal to the specified level are printed. The number of printed logs decreases as the specified log level increases.

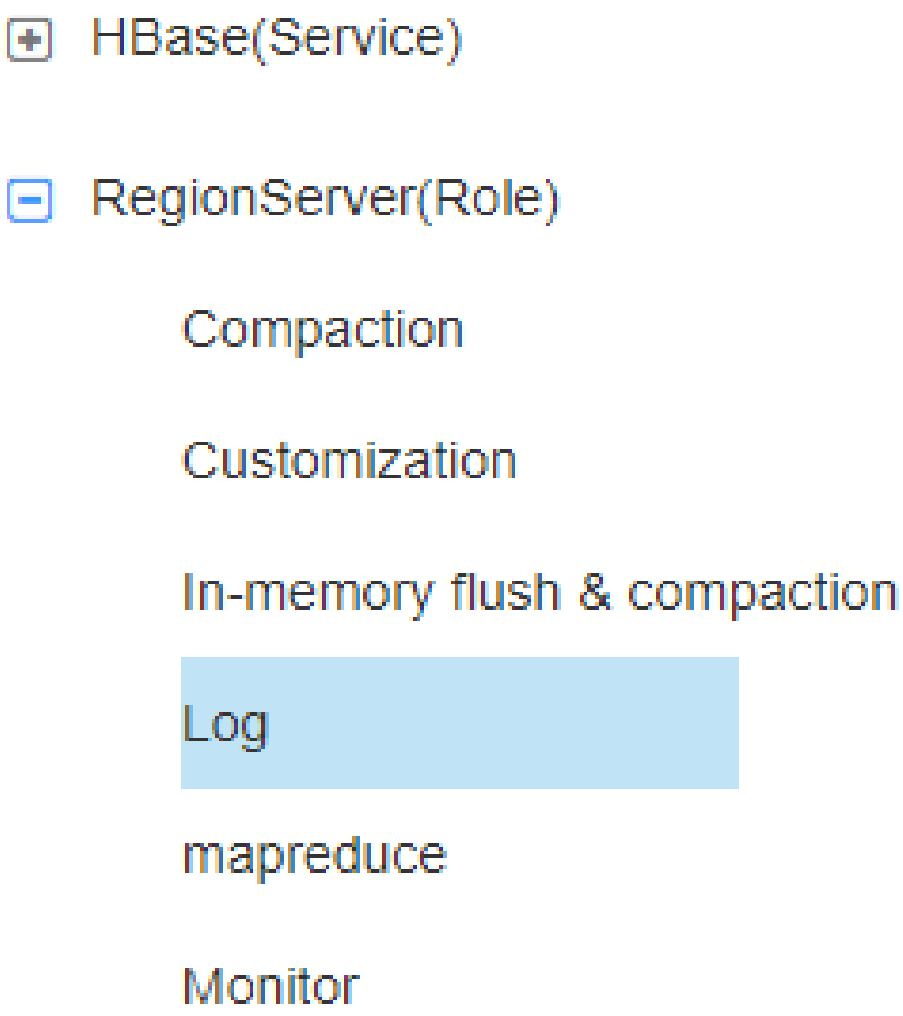
Table 9-30 Log levels

Level	Description
FATAL	Logs of this level record fatal error information about the current event processing that may result in a system crash.
ERROR	Logs of this level record error information about the current event processing, which indicates that system running is abnormal.
WARN	Logs of this level record abnormal information about the current event processing. These abnormalities will not result in system faults.
INFO	Logs of this level record normal running status information about the system and events.
DEBUG	Logs of this level record the system information and system debugging information.

To modify log levels, perform the following operations:

Step 1 Go to the **All Configurations** page of the HBase service. For details, see **Modifying Cluster Service Configuration Parameters**.

Step 2 On the left menu bar, select the log menu of the target role.



Step 3 Select a desired log level.

Step 4 Save the configuration. In the displayed dialog box, click **OK** to make the configurations take effect.

NOTE

The configurations take effect immediately without the need to restart the service.

----End

Log Formats

The following table lists the HBase log formats.

Table 9-31 Log formats

Type	Component	Format	Example
Run logs	HMaster	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<*Thread that generates the log*>|<*Message in the log*>|<*Location of the log event*>	2020-01-19 16:04:53,558 | INFO | main | env:HBASE_THRIFT_OPTS= | org.apache.hadoop.hbase.util.ServerCommandLine.logProcessInfo(ServerCommandLine.java:113)
	RegionServer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<*Thread that generates the log*>|<*Message in the log*>|<*Location of the log event*>	2020-01-19 16:05:18,589 | INFO | regionserver16020-SendThread(linux-k6da:2181) | Client will use GSSAPI as SASL mechanism. | org.apache.zookeeper.client.ZooKeeperSaslClient$1.run(ZooKeeperSaslClient.java:285)
	ThriftServer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<*Thread that generates the log*>|<*Message in the log*>|<*Location of the log event*>	2020-02-16 09:42:55,371 | INFO | main | loaded properties from hadoop-metrics2.properties | org.apache.hadoop.metrics2.impl.MetricsConfig.loadFirst(MetricsConfig.java:111)
Audit logs	HMaster	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<*Thread that generates the log*>|<*Message in the log*>|<*Location of the log event*>	2020-02-16 09:42:40,934 | INFO | master:linux-k6da:16000 | Master: [master:linux-k6da:16000] start operation called. | org.apache.hadoop.hbase.master.HMaster.run(HMaster.java:581)
	RegionServer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<*Thread that generates the log*>|<*Message in the log*>|<*Location of the log event*>	2020-02-16 09:42:51,063 | INFO | main | RegionServer: [regionserver16020] start operation called. | org.apache.hadoop.hbase.regionserver.HRegionServer.startRegionServer(HRegionServer.java:2396)

Type	Component	Format	Example
	ThriftServer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<*Thread that generates the log*>|<*Message in the log*>|<*Location of the log event*>	2020-02-16 09:42:55,512 | INFO | main | thrift2 server start operation called. | org.apache.hadoop.hbase.thrift2.ThriftServer.main(ThriftServer.java:421)

9.8.2 Configuring Region In Transition Recovery Chore Service

Scenario

In a faulty environment, there are possibilities that a region may be stuck in transition for longer duration due to various reasons like slow region server response, unstable network, ZooKeeper node version mismatch. During region transition, client operation may not work properly as some regions will not be available.

Enabling the Region Transition Recovery Function

A chore service should be scheduled at HMaster to identify and recover regions that stay in the transition state for a long time.

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HBase** > **Configurations**. The following table describes the parameters for enabling this function.

Table 9-32 Parameters

Parameter	Description	Default Value
hbase.region.assignment.auto.recovery.enabled	Configuration parameter used to enable/disable the region assignment recovery thread feature.	true

9.8.3 Enabling Inter-Cluster Copy to Back Up Data

Scenario

DistCp is used to copy the data stored on HDFS from a cluster to another cluster. DistCp depends on the cross-cluster copy function, which is disabled by default. This function needs to be enabled in both clusters.

Modify parameters on MRS to enable cross-cluster copy.

Impact on the System

Yarn needs to be restarted to enable the cross-cluster copy function and cannot be accessed during the restart.

Prerequisites

The **hadoop.rpc.protection** parameter of the two HDFS clusters must be set to the same data transmission mode, which can be **privacy** (encryption enabled) or **authentication** (encryption disabled).

NOTE

You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** > **Configurations**, and search for **hadoop.rpc.protection**.

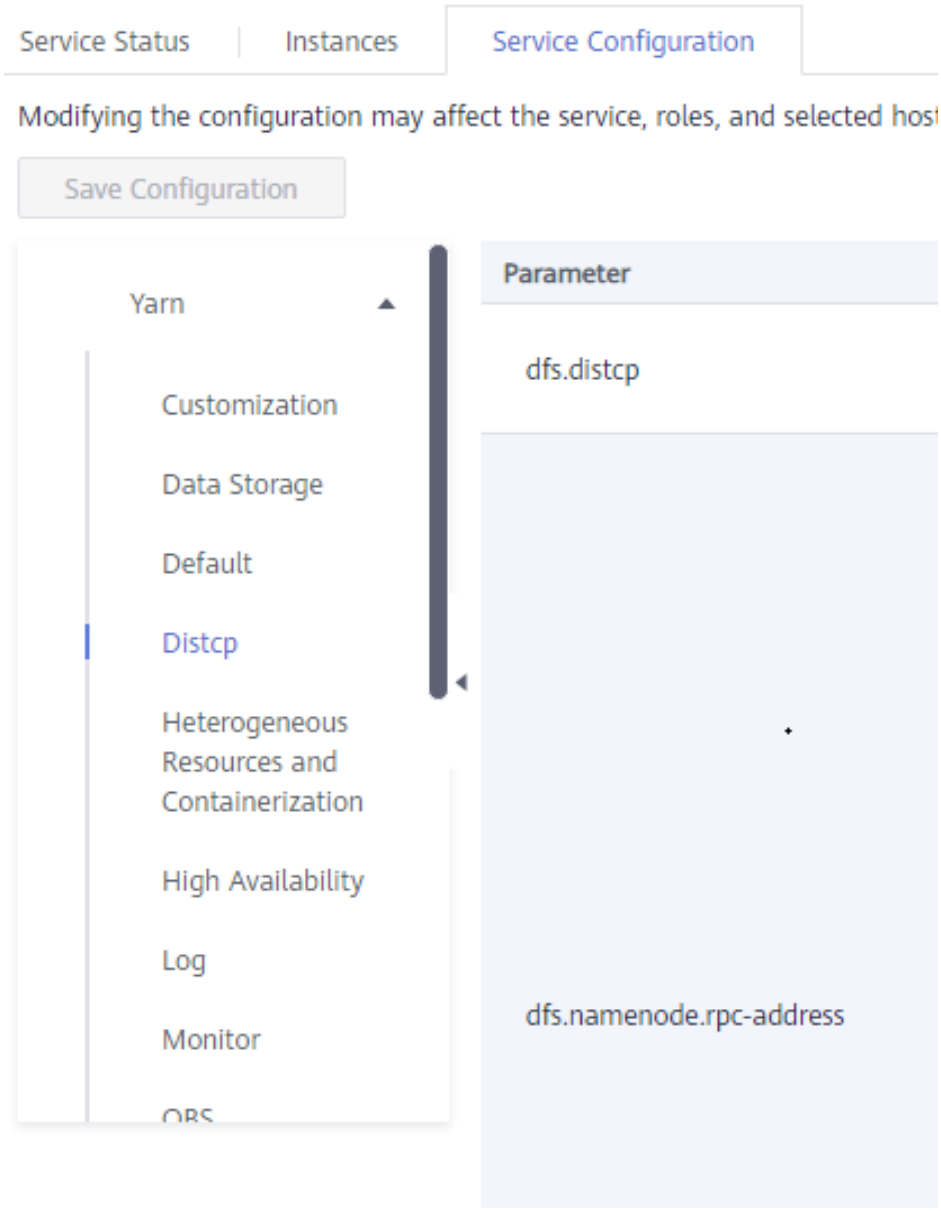
Procedure

Step 1 Go to the **All Configurations** page of the Yarn service. For details, see **Modifying Cluster Service Configuration Parameters**.

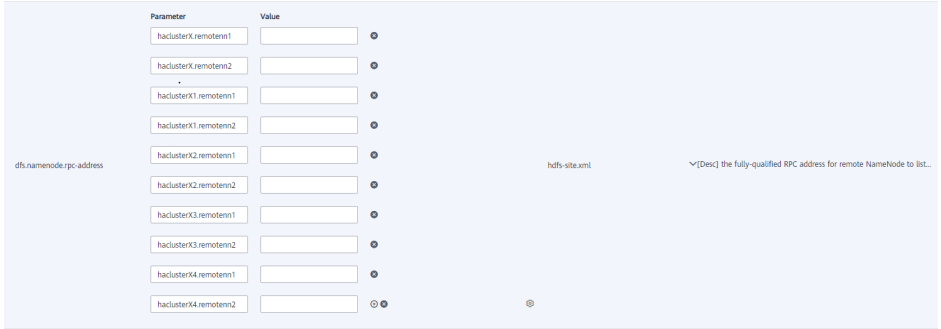
NOTE

If the **Components** tab is unavailable, complete IAM user synchronization first. (On the **Dashboard** page, click **Synchronize** on the right side of **IAM User Sync** to synchronize IAM users.)

Step 2 In the navigation pane, choose **Yarn** > **Distcp**.



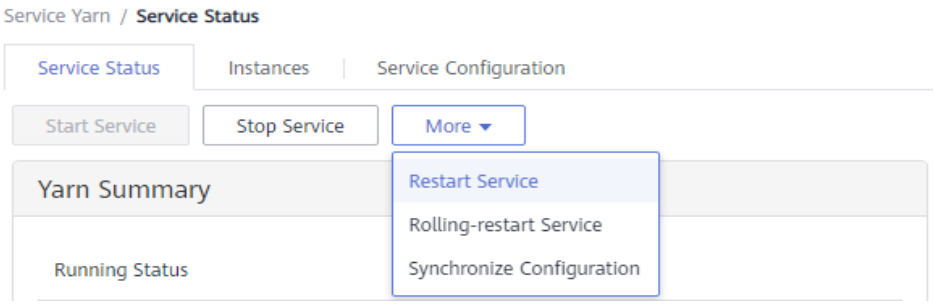
Step 3 Set **haclusterX.remotenn1** of **dfs.namenode.rpc-address** to the service IP address and RPC port number of one NameNode instance of the peer cluster, and set **haclusterX.remotenn2** to the service IP address and RPC port number of the other NameNode instance of the peer cluster. Enter a value in the *IP address:port* format.



dfs.namenode.rpc-address.haclusterX.remotenn1 and **dfs.namenode.rpc-address.haclusterX.remotenn2** do not distinguish active and standby NameNode instances. To obtain the service IP address of the NameNode instance, log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS** > **Instances**. The NameNode RPC port can be obtained by searching for **dfs.namenode.rpc.port** on the HDFS service **configurations** page. It cannot be changed on Manager. The port cannot be changed on FusionInsight Manager.

For example, **10.1.1.1:9820** and **10.1.1.2:9820**.

Step 4 Save the configuration. On the **Dashboard** tab page, and choose **More** > **Restart Service** to restart the Yarn service.



Operation succeeded is displayed. Click **Finish**. The Yarn service is started successfully.

Step 5 Log in to the other cluster and repeat the preceding operations.

----End

9.8.4 Configuring Automatic Data Backup for Active and Standby HBase Clusters

Prerequisites

- Active and standby clusters have been installed and started.
- Time is consistent between the active and standby clusters and the NTP service on the active and standby clusters uses the same time source.
- When the HBase service of the active cluster is stopped, the ZooKeeper and HDFS services must be started and run.
- ReplicationSyncUp must be run by the system user who starts the HBase process.
- In security mode, ensure that the HBase system user of the standby cluster has the read permission on HDFS of the active cluster. This is because that it will update the ZooKeeper nodes and HDFS files of the HBase system.
- When HBase of the active cluster is faulty, the ZooKeeper, file system, and network of the active cluster are still available.

Scenarios

The replication mechanism can use WAL to synchronize the state of a cluster with the state of another cluster. After HBase replication is enabled, if the active cluster is faulty, ReplicationSyncUp synchronizes incremental data from the active cluster

to the standby cluster using the information from the ZooKeeper node. After data synchronization is complete, the standby cluster can be used as an active cluster.

Parameter Configuration

Parameter	Description	Default Value
hbase.replication.bulkload.enabled	Whether to enable the bulkload data replication function. The parameter value type is Boolean. To enable the bulkload data replication function, set this parameter to **true** for the active cluster.	**false**
hbase.replication.cluster.id	ID of the source HBase cluster. After the bulkload data replication is enabled, this parameter is mandatory and must be defined in the source cluster. The parameter value type is String.	-

Using the ReplicationSyncUp Tool

Run the following command on the hbase shell of the active cluster:

hbase org.apache.hadoop.hbase.replication.regionserver.ReplicationSyncUp -Dreplication.sleep.before.failover=1

NOTE

replication.sleep.before.failover indicates sleep time required for replication of the remaining data when RegionServer fails to start. You are advised to set this parameter to 1 second to quickly trigger replication.

Precautions

- When the active cluster is stopped, this tool obtains the WAL processing progress and WAL processing queue from the ZooKeeper Node (RS znode) and copies the queues that are not copied to the standby cluster.
- RegionServer of each active cluster has its own znode under the replication node of ZooKeeper in the standby cluster. It contains one znode of each peer cluster.
- If RegionServer is faulty, each RegionServer in the active cluster receives a notification through the watcher and attempts to lock the znode of the faulty RegionServer, including its queues. The successfully created RegionServer transfers all queues to the znode of its own queue. After queues are transferred, they are deleted from the old location.
- When the active cluster is stopped, ReplicationSyncUp synchronizes data between active and standby clusters using the information from the ZooKeeper node. In addition, WALs of the RegionServer znode will be moved to the standby cluster.

Restrictions and Limitations

If the standby cluster is stopped or the peer relationship is closed, the tool runs normally but the peer relationship cannot be replicated.

9.8.5 Configuring HBase Cluster HA and DR

9.8.5.1 Configuring HBase Active/Standby DR

Scenario

HBase disaster recovery (DR), a key feature that is used to ensure high availability (HA) of the HBase cluster system, provides the real-time remote DR function for HBase. HBase DR provides basic O&M tools, including tools for maintaining and re-establishing DR relationships, verifying data, and querying data synchronization progress. To implement real-time DR, back up data of an HBase cluster to another HBase cluster. DR in the HBase table common data writing and BulkLoad batch data writing scenarios is supported.

Prerequisites

- The active and standby clusters are successfully installed and started, and you have the administrator permissions on the clusters.
- Ensure that the network connection between the active and standby clusters is normal and ports are available.
- If the active cluster is deployed in security mode and is not managed by one FusionInsight Manager, cross-cluster trust relationship has been configured for the active and standby clusters.. If the active cluster is deployed in normal mode, no cross-cluster mutual trust is required.
- Cross-cluster replication has been configured for the active and standby clusters.
- Time is consistent between the active and standby clusters and the NTP service on the active and standby clusters uses the same time source.
- The mapping between host names and service IP addresses of all nodes in the active and standby clusters have been configured in the **hosts** file of these nodes.

  NOTE

  If the client of the active cluster is installed on a node outside the cluster, the mapping between host names and service IP addresses of all nodes in the active and standby clusters must have been configured in the **hosts** file of these nodes.
- The network bandwidth between the active and standby clusters is determined based on service volume, which cannot be less than the possible maximum service volume.
- The MRS versions of the active and standby clusters must be the same.
- The scale of the standby cluster must be greater than or equal to that of the active cluster.

Constraints

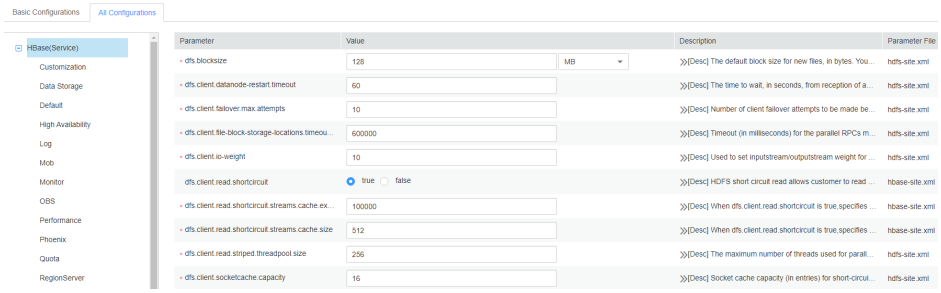
- Although DR provides the real-time data replication function, the data synchronization progress is affected by many factors, such as the service volume in the active cluster and the health status of the standby cluster. In normal cases, the standby cluster should not take over services. In extreme cases, system maintenance personnel and other decision makers determine whether the standby cluster takes over services according to the current data synchronization indicators.
- HBase clusters must be deployed in active/standby mode.
- Table-level operations on the DR table of the standby cluster are forbidden, such as modifying the table attributes and deleting the table. Misoperations on the standby cluster will cause data synchronization failure of the active cluster. As a result, table data in the standby cluster is lost.
- If the DR data synchronization function is enabled for HBase tables of the active cluster, the DR table structure of the standby cluster needs to be modified to ensure table structure consistency between the active and standby clusters during table structure modification.

Procedure

Configuring the common data writing DR parameters for the active cluster

Step 1 Log in to Manager of the active cluster.

Step 2 Choose **Cluster** > **Services** > **HBase** > **Configurations** and click **All Configurations**. The HBase configuration page is displayed.



Step 3 (Optional) **Table 9-33** describes the optional configuration items during HBase DR. You can set the parameters based on the description or use the default values.

Table 9-33 Optional configuration items

Navigation Path	Parameter	Default Value	Description
HMaster > Performance	hbase.master.logcleaner.ttl	600000	Specifies the retention period of HLog. If the value is set to **604800000** (unit: millisecond), the retention period of HLog is 7 days.

Navigation Path	Parameter	Default Value	Description
	hbase.master.cleaner.interval	60000	Interval for the HMaster to delete historical HLog files. The HLog that exceeds the configured period will be automatically deleted. You are advised to set it to the maximum value to save more HLogs.
RegionServer > Replication	replication.source.size.capacity	16777216	Maximum size of edits, in bytes. If the edit size exceeds the value, HLog edits will be sent to the standby cluster.
	replication.source.nb.capacity	25000	Maximum number of edits, which is another condition for triggering HLog edits to be sent to the standby cluster. After data in the active cluster is synchronized to the standby cluster, the active cluster reads and sends data in HLog according to this parameter value. This parameter is used together with **replication.source.size.capacity**.
	replication.source.maxretriesmultiplier	10	Maximum number of retries when an exception occurs during replication.
	replication.source.sleepforretries	1000	Retry interval (Unit: ms)
	hbase.regionserver.replication.handler.count	6	Number of replication RPC server instances on RegionServer

Configuring the BulkLoad batch data writing DR parameters for the active cluster

Step 4 Determine whether to enable the BulkLoad batch data writing DR function.

If yes, go to **Step 5**.

If no, go to **Step 8**.

Step 5 Choose **Cluster** > **Services** > **HBase** > **Configurations** and click **All Configurations**. The HBase configuration page is displayed.

Step 6 Search for **hbase.replication.bulkload.enabled** and change its value to **true** to enable the BulkLoad batch data writing DR function.

Step 7 Search for **hbase.replication.cluster.id** and change the HBase ID of the active cluster. The ID is used by the standby cluster to connect to the active cluster. The value can contain uppercase letters, lowercase letters, digits, and underscores (_), and cannot exceed 30 characters.

Restarting the HBase service and install the client

Step 8 Click **Save**. Click **Dashboard**, choose **More** > **Restart Service**, enter the password of the current user, and click OK to restart the HBase service.

Step 9 In the active and standby clusters, choose **Cluster** > **Service** > **HBase** > **More** > **Download Client** to download the client and install it. For details, see **Installing a Client**.

Adding the DR relationship between the active and standby clusters

Step 10 Log in to the node where the client of the active cluster is installed as the client installation user and log in to the HBase shell page as the **hbase** user.

cd *Client installation directory*

source bigdata_env

kinit hbase (Skip this operation for clusters where Kerberos authentication is not enabled.)

hbase shell

Step 11 Run the following command on **hbase shell** to create the DR synchronization relationship between the active cluster HBase and the standby cluster HBase.

add_peer '*Standby cluster ID***', CLUSTER_KEY => "***ZooKeeper service IP address in the standby cluster***", CONFIG => {"hbase.regionserver.kerberos.principal" => "***Standby cluster RegionServer principal***", "hbase.master.kerberos.principal" => "***Standby cluster HMaster principal***"}**

- The standby cluster ID indicates the ID for the active cluster to recognize the standby cluster. Enter an ID. The value can be specified randomly. Digits are recommended.
- The ZooKeeper address of the standby cluster includes the service IP address of ZooKeeper, the port for listening to client connections, and the HBase root directory of the standby cluster on ZooKeeper.
  - To view the service IP address of the ZooKeeper instance in the standby cluster, choose **Cluster** > **Services** > **ZooKeeper** > **Instances** on FusionInsight Manager.
  - Port for the ZooKeeper client of the standby cluster to connect to the server and root directory of the HBase of the standby cluster on ZooKeeper. You can log in to FusionInsight Manager and choose **Cluster** > **Services** > **ZooKeeper** > **Configurations**. Search for the **clientPort** and **zookeeper.znode.parent** parameters.
- Search for **hbase.master.kerberos.principal** and **hbase.regionserver.kerberos.principal** in the *Client installation directory*/**HBase/hbase/conf/hbase-site.xml** configuration file of the standby cluster.

For example, to add the DR relationship between the active and standby clusters, run the **add_peer '***Standby cluster ID***', CLUSTER_KEY =>**

"*192.168.40.2,192.168.40.3,192.168.40.4:24002:/hbase***", CONFIG => {"hbase.regionserver.kerberos.principal" => "***hbase/hadoop.hadoop.com@HADOOP.COM***", "hbase.master.kerberos.principal" => "***hbase/hadoop.hadoop.com@HADOOP.COM***"}**

Step 12 (Optional) If the BulkLoad batch data write DR function is enabled, the HBase client configuration of the active cluster must be copied to the standby cluster.

1. Create the **/hbase/replicationConf/***hbase.replication.cluster.id* ***of the active cluster*** directory in the HDFS of the standby cluster.
2. HBase client configuration file, which is copied to the **/hbase/replicationConf/***hbase.replication.cluster.id* ***of the active cluster*** directory of the HDFS of the standby cluster. For Example:

   hdfs dfs -put HBase/hbase/conf/core-site.xml HBase/hbase/conf/hdfs-site.xml HBase/hbase/conf/yarn-site.xml hdfs://*NameNode IP.25000***/hbase/replicationConf/*** ***source_cluster***

Enabling HBase DR to synchronize data

Step 13 Check whether a naming space exists in the HBase service instance of the standby cluster and the naming space has the same name as the naming space of the HBase table for which the DR function is to be enabled.

- If the same namespace exists, go to **Step 14**.
- If no, create a naming space with the same name in the HBase shell of the standby cluster and go to **Step 14**.

Step 14 In the HBase shell of the active cluster, run the following command as user **hbase** to enable the real-time DR function for the table data of the active cluster to ensure that the data modified in the active cluster can be synchronized to the standby cluster in real time.

You can only synchronize the data of one HTable at a time.

enable_table_replication '*table name***'**

NOTE

- If the standby cluster does not contain a table with the same name as the table for which real-time synchronization is to be enabled, the table is automatically created.
- If a table with the same name as the table for which real-time synchronization is to be enabled exists in the standby cluster, the structures of the two tables must be the same.
- If the encryption algorithm SMS4 or AES is configured for '*Table name*', the function for synchronizing data from the active cluster to the standby cluster cannot be enabled for the HBase table.
- If the standby cluster is offline or has tables with the same name but different structures, the DR function cannot be enabled.
- If the DR data synchronization function is enabled for some Phoenix tables in the active cluster, the standby cluster cannot have common HBase tables with the same names as the Phoenix tables in the active cluster. Otherwise, the DR function fails to be enabled or the tables with the names in the standby cluster cannot be used properly.
- If the DR data synchronization function is enabled for Phoenix tables in the active cluster, you need to enable the DR data synchronization function for the metadata tables of the Phoenix tables. The metadata tables include SYSTEM.CATALOG, SYSTEM.FUNCTION, SYSTEM.SEQUENCE, and SYSTEM.STATS.
- If the DR data synchronization function is enabled for HBase tables of the active cluster, after adding new indexes to HBase tables, you need to manually add secondary indexes to DR tables in the standby cluster to ensure secondary index consistency between the active and standby clusters.

Step 15 (Optional) If HBase does not use Ranger, run the following command as user **hbase** in the HBase shell of the active cluster to enable the real-time permission to control data DR function for the HBase tables in the active cluster.

enable_table_replication 'hbase:acl'

Creating Users

Step 16 Log in to FusionInsight Manager of the standby cluster, choose **System** > **Permission** > **Role** > **Create Role** to create a role, and add the same permission for the standby data table to the role based on the permission of the HBase source data table of the active cluster.

Step 17 Choose **System** > **Permission** > **User** > **Create** to create a user. Set the **User Type** to **Human-Machine** or **Machine-Machine** based on service requirements and add the user to the created role. Access the HBase DR data of the standby cluster as the newly created user.

NOTE

- After the permission of the active HBase source data table is modified, to ensure that the standby cluster can properly read data, modify the role permission for the standby cluster.
- If the current component uses Ranger for permission control, you need to configure permission management policies based on Ranger. For details, see **Adding a Ranger Access Permission Policy for HBase**.

Synchronizing the table data of the active cluster

Step 18 After HBase DR is configured and data synchronization is enabled, check whether tables and data exist in the active cluster and whether the historical data needs to be synchronized to the standby cluster.

- If yes, a table exists and data needs to be synchronized. Log in as the HBase table user to the node where the HBase client of the active cluster is installed

and run the kinit username to authenticate the identity. The user must have the read and write permissions on tables and the execute permission on the **hbase:meta** table. Then go to **Step 19**.

- If no, no further action is required.

Step 19 The HBase DR configuration does not support automatic synchronization of historical data in tables. You need to back up the historical data of the active cluster and then manually restore the historical data in the standby cluster.

Manual recovery refers to the recovery of a single table, which can be performed through Export, DistCp, or Import.

To manually recover a single table, perform the following steps:

1. Export table data from the active cluster.

   hbase org.apache.hadoop.hbase.mapreduce.Export -Dhbase.mapreduce.include.deleted.rows=true *Table name Directory where the source data is stored*

   Example: **hbase org.apache.hadoop.hbase.mapreduce.Export -Dhbase.mapreduce.include.deleted.rows=true t1 /user/hbase/t1**

2. Copy the data that has been exported to the standby cluster.

   hadoop distcp *directory where the source data is stored on the active cluster* **hdfs://***ActiveNameNodeIP:8020/directory where the source data is stored on the standby cluster*

   ActiveNameNodeIP indicates the IP address of the active NameNode in the standby cluster.

   Example: **hadoop distcp /user/hbase/t1 hdfs://192.168.40.2:8020/user/hbase/t1**

3. Import data to the standby cluster as the HBase table user of the standby cluster.

   On the HBase shell screen of the standby cluster, run the following command as user **hbase** to retain the data writing status:

   set_clusterState_active

   The command is run successfully if the following information is displayed:

   ```
   hbase(main):001:0> set_clusterState_active
   => true
   ```

   hbase org.apache.hadoop.hbase.mapreduce.Import *-Dimport.bulk.output=Directory where the output data is stored in the standby cluster Table name Directory where the source data is stored in the standby cluster*

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *Directory where the output data is stored in the standby cluster Table name*

   Example:

   ```
   hbase(main):001:0> set_clusterState_active
   => true
   ```

   hbase org.apache.hadoop.hbase.mapreduce.Import -Dimport.bulk.output=/user/hbase/output_t1 t1 /user/hbase/t1

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /user/hbase/output_t1 t1

Step 20 Run the following command on the HBase client to check the synchronized data of the active and standby clusters. After the DR data synchronization function is enabled, you can run this command to check whether the newly synchronized data is consistent.

hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication --starttime=*Start time* **--endtime**=*End time Column family name ID of the standby cluster Table name*

NOTE

- The start time must be earlier than the end time.
- The values of **starttime** and **endtime** must be in the timestamp format. You need to run **date -d "2015-09-30 00:00:00" +%s to** change a common time format to a timestamp format.

Specify the data writing status for the active and standby clusters.

Step 21 On the **hbase shell** screen of the active cluster, run the following command as user **hbase** to retain the data writing status:

set_clusterState_active

The command is run successfully if the following information is displayed:

```
hbase(main):001:0> set_clusterState_active
=> true
```

Step 22 On the **hbase shell** screen of the standby cluster, run the following command as user **hbase** to retain the data read-only status:

set_clusterState_standby

The command is run successfully if the following information is displayed:

```
hbase(main):001:0> set_clusterState_standby
=> true
```

----End

Related Commands

Table 9-34 HBase DR

Operation	Command	Description
Set up a DR relationship.	add_peer'*Standby cluster ID*', CLUSTER_KEY => "*Standby cluster ZooKeeper service IP address*", CONFIG => {"hbase.regionserver.kerberos.principal" => "*Standby cluster RegionServer principal*", "hbase.master.kerberos.principal" => "*Standby cluster HMaster principal*"} **add_peer '1','zk1,zk2,zk3:2181:/hbase1'** **2181**: port number of ZooKeeper in the cluster	Set up the relationship between the active cluster and the standby cluster. If BulkLoad batch data write DR is enabled: 1. Create the **/hbase/replicationConf/***hbase.replication.cluster.id* ***of the active cluster*** directory in the HDFS of the standby cluster. 2. HBase client configuration file, which is copied to the **/hbase/replicationConf/***hbase.replication.cluster.id* ***of the active cluster*** directory of the HDFS of the standby cluster.
Remove the DR relationship.	**remove_peer** *'Standby cluster ID'* Example: **remove_peer '1'**	Remove standby cluster information from the active cluster.
Querying the DR Relationship	**list_peers**	Query standby cluster information (mainly Zookeeper information) in the active cluster.
Enable the real-time user table synchronization function.	**enable_table_replication** *'Table name'* Example: **enable_table_replication 't1'**	Synchronize user tables from the active cluster to the standby cluster.
Disable the real-time user table synchronization function.	**disable_table_replication** *'Table name'* Example: **disable_table_replication 't1'**	Do not synchronize user tables from the active cluster to the standby cluster.

Operation	Command	Description
Verify data of the active and standby clusters.	**bin/hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication --starttime=***Start time* **--endtime=***End time Column family name Standby cluster ID Table name*	Verify whether data of the specified table is the same between the active cluster and the standby cluster. The description of the parameters in this command is as follows: ● Start time: If start time is not specified, the default value **0** will be used. ● End time: If end time is not specified, the time when the current operation is submitted will be used by default. ● Table name: If a table name is not entered, all user tables for which the real-time synchronization function is enabled will be verified by default.
Switch the data writing status.	**set_clusterState_active** **set_clusterState_standby**	Specifies whether data can be written to the cluster HBase tables.

Operation	Command	Description
Add or update the active cluster HDFS configurations saved in the peer cluster.	**hdfs dfs -put -f HBase/hbase/conf/core-site.xml HBase/hbase/conf/hdfs-site.xml HBase/hbase/conf/yarn-site.xml hdfs://***Standby cluster* ***NameNode IP:PORT*****/hbase/replicationConf/***Active cluster* ***hbase.replication.cluster.id***	Enable DR for data including bulkload data. When HDFS parameters are modified in the active cluster, the modification cannot be automatically synchronized from the active cluster to the standby cluster. You need to manually run the command to synchronize configuration. The affected parameters are as follows: • fs.defaultFS • dfs.client.failover.proxy.provider.hacluster • dfs.client.failover.connection.retries.on.timeouts • dfs.client.failover.connection.retries For example, change **fs.defaultFS** to **hdfs://hacluster_sale**, HBase client configuration file, which is copied to the **/hbase/replicationConf/*hbase.replication.cluster.id of the active cluster*** directory of the HDFS of the standby cluster.
Disable replication for bulk loaded data of a single table, when **hbase.replication.bulkload.enabled** is set to **true**.	**disable_bulkload_replication** *'peerId', 'Table name'* Example: **disable_bulkload_replication '1','t1'**	If replication for bulk loaded data is enabled for the active cluster and real-time synchronization is enabled for a table, you can run this command to pause replication for bulk loaded data of the table. Run the **list_peers** command to view *peerId* in the standby cluster information. This parameter is available only in MRS 3.3.0 or later.

Operation	Command	Description
Enable replication for bulk loaded data of a single table, when **hbase.replication.bulkload.enabled** is set to **true**.	**enable_bulkload_replication 'peerId', 'Table name'** Example: **enable_bulkload_replication '1','t1'**	Replication for bulk loaded data is enabled for the active cluster, and real-time synchronization is enabled for a table. You can run this command to resume replication for bulk loaded data of the table after the function is paused. Run the **get_peer_config** *'peerId'* command to check the DR configuration of the standby cluster. If the value of a table name field that starts with **BULREP_** is **false**, replication for bulk loaded data is disabled for the table. **NOTE** • This parameter is available only in MRS 3.3.0 or later. • Data generated when replication for bulk loaded data is paused will not be synchronized after the function is resumed. • It takes several minutes for this operation to take effect on the server. To prevent bulk loaded data loss caused by synchronization latency, ensure that "Update peer configs succeed." is printed in run logs on each RegionServer after this command is executed. Then, perform subsequent operations on the active cluster.

9.8.5.2 Switching Between Active and Standby HBase Clusters

Scenario

The HBase cluster in the current environment is a DR cluster. Due to some reasons, the active and standby clusters need to be switched over. That is, the standby cluster becomes the active cluster, and the active cluster becomes the standby cluster.

Impact on the System

After the active and standby clusters are switched over, data cannot be written to the original active cluster, and the original standby cluster becomes the active cluster to take over upper-layer services.

Procedure

Ensuring that upper-layer services are stopped

Step 1 Ensure that the upper-layer services have been stopped. If not, perform operations by referring to **Configuring HBase Standby Cluster Information for Switchover**.

Disabling the write function of the active cluster

Step 2 Download and install the HBase client.

For details, see **Installing a Client**.

Step 3 On the HBase client of the standby cluster, run the following command as user **hbase** to disable the data write function of the standby cluster:

cd *Client installation directory*

source bigdata_env

kinit hbase

hbase shell

set_clusterState_standby

The command is run successfully if the following information is displayed:

```
hbase(main):001:0> set_clusterState_standby
=> true
```

Checking whether the active/standby synchronization is complete

Step 4 Run the following command to ensure that the current data has been synchronized (SizeOfLogQueue=0 and SizeOfLogToReplicate=0 are required). If the values are not 0, wait and run the following command repeatedly until the values are 0.

status 'replication'

Disabling synchronization between the active and standby clusters

Step 5 Query all synchronization clusters and obtain the value of **PEER_ID**.

list_peers

Step 6 Delete all synchronization clusters.

remove_peer *'Standby cluster ID'*

Example:

remove_peer *'1'*

Step 7 Query all synchronized tables.

list_replicated_tables

Step 8 Disable all synchronized tables queried in the **Step 7** step.

disable_table_replication *'Table name'*

Example:

disable_table_replication *'t1'*

Performing an active/standby switchover

Step 9 Reconfigure HBase DR. For details, see **Configuring HBase Active/Standby DR**.

----End

9.8.5.3 Configuring HBase Standby Cluster Information for Switchover

Scenario

MRS cluster administrators can configure HBase cluster DR to improve system availability. If the active cluster in the DR environment is faulty and the connection to the HBase upper-layer application is affected, you need to configure the standby cluster information for the HBase upper-layer application so that the application can run in the standby cluster.

Impact on the System

After a service switchover, data written to the standby cluster is not synchronized to the active cluster by default. Add the active cluster is recovered, the data newly generated in the standby cluster needs to be synchronized to the active cluster by backup and recovery. If automatic data synchronization is required, you need to switch over the active and standby HBase DR clusters.

Procedure

Step 1 Log in to FusionInsight Manager of the standby cluster.

Step 2 Download and install the HBase client.

Step 3 On the HBase client of the standby cluster, run the following command as user **hbase** to enable the data writing status in the standby cluster.

cd *Client installation directory*

source bigdata_env

kinit hbase

hbase shell

set_clusterState_active

The command is run successfully if the following information is displayed:

```
hbase(main):001:0> set_clusterState_active
=> true
```

Step 4 Check whether the original configuration files **hbase-site.xml**, **core-site.xml**, and **hdfs-site.xml** of the HBase upper-layer application are modified to adapt to the application running.

- If yes, update the related content to the new configuration file and replace the old configuration file.
- If no, use the new configuration file to replace the original configuration file of the HBase upper-layer application.

Step 5 Configure the network connection between the host where the HBase upper-layer application is located and the standby cluster.

NOTE

If the host where the client is installed is not a node in the cluster, configure network connections for the client to prevent errors when you run commands on the client.

1. Ensure that the host where the client is installed can communicate with the hosts listed in the **hosts** file in the directory where the client installation package is decompressed.
2. If the host where the client is located is not a node in the cluster, you need to set the mapping between the host name and the IP address (service plan) in the /etc/hosts file on the host. The host names and IP addresses must be mapped one by one.

Step 6 Set the time of the host where the HBase upper-layer application is located to be the same as that of the standby cluster. The time difference must be less than 5 minutes.

Step 7 Check the authentication mode of the active cluster.

- If the security mode is used, go to **Step 8**.
- If the normal mode is used, no further action is required.

Step 8 Obtain the **keytab** and **krb5.conf** configuration files of the HBase upper-layer application user.

1. On FusionInsight Manager of the standby cluster, choose **System** > **Permission** > **User**.
2. Locate the row that contains the target user, click **More** > **Download Authentication Credential** in the **Operation** column, and download the **keytab** file to the local PC.
3. Decompress the package to obtain **user.keytab** and **krb5.conf**.

Step 9 Use the **user.keytab** and **krb5.conf** files to replace the original files in the HBase upper-layer application.

Step 10 Stop upper-layer applications.

Step 11 Determine whether to switch over the active and standby HBase clusters. If the switchover is not performed, data will not be synchronized.

- If yes, switch over the active and standby HBase DR clusters. For details, see **Switching Between Active and Standby HBase Clusters**. Then, go to **Step 12**.
- If no, go to **Step 12**.

Step 12 Start the upper-layer services.

----End

9.9 Common Issues About HBase

9.9.1 Operation Failures Occur in Stopping BulkLoad On the Client

Question

Why submitted operations fail by stopping BulkLoad on the client during BulkLoad data importing?

Answer

When BulkLoad is enabled on the client, a partitioner file is generated and used to demarcate the range of Map task data inputting.

The file is automatically deleted when BulkLoad exists on the client.

In general, if all map tasks are enabled and running, the termination of BulkLoad on the client does not cause the failure of submitted operations.However, due to the retry and speculative execution mechanism of Map tasks, a Map task is performed again if failures of the Reduce task to download the data of the completed Map task exceed the limit. In this case, if BulkLoad already exists on the client, the retry Map task fails and the operation failure occurs because the partitioner file is missing.

Therefore, it is recommended not to stop BulkLoad on the client during BulkLoad data importing.

9.9.2 How Do I Restore a Region in the RIT State for a Long Time?

Question

How do I restore a region in the RIT state for a long time?

Answer

Log in to the HMaster Web UI, choose **Procedure & Locks** in the navigation tree, and check whether any process ID is in the **Waiting** state. If yes, run the following command to release the procedure lock:

hbase hbck -j *Client installation directory*/**HBase/hbase/tools/hbase-hbck2-*.jar bypass -o** *pid*

Check whether the state is in the **Bypass** state. If the procedure on the UI is always in **RUNNABLE(Bypass)** state, perform an active/standby switchover. Run the **assigns** command to bring the region online again.

hbase hbck -j *Client installation directory*/**HBase/hbase/tools/hbase-hbck2-*.jar assigns -o** *regionName*

9.9.3 Why Does HMaster Exits Due to Timeout When Waiting for the NameSpace Table to Go Online?

Question

Why does HMaster exit due to timeout when waiting for the namespace table to go online?

Answer

During the HMaster active/standby switchover or startup, HMaster performs WAL splitting and region recovery for the RegionServer that failed or was stopped previously.

Multiple threads are running in the background to monitor the HMaster startup process.

- TableNamespaceManager

  This is a help class, which is used to manage the allocation of namespace tables and monitoring table regions during HMaster active/standby switchover or startup. If the namespace table is not online within the specified time (**hbase.master.namespace.init.timeout**, which is 3,600,000 ms by default), the thread terminates HMaster abnormally.

- InitializationMonitor

  This is an initialization thread monitoring class of the primary HMaster, which is used to monitor the initialization of the primary HMaster. If a thread fails to be initialized within the specified time (**hbase.master.initializationmonitor.timeout**, which is 3,600,000 ms by default), the thread terminates HMaster abnormally. If **hbase.master.initializationmonitor.haltontimeout** is started, the default value is **false**.

During the HMaster active/standby switchover or startup, if the **WAL hlog** file exists, the WAL splitting task is initialized. If the WAL hlog splitting task is complete, it initializes the table region allocation task.

HMaster uses ZooKeeper to coordinate log splitting tasks and valid RegionServers and track task development. If the primary HMaster exits during the log splitting task, the new primary HMaster attempts to resend the unfinished task, and RegionServer starts the log splitting task from the beginning.

The initialization of the HMaster is delayed due to the following reasons:

- Network faults occur intermittently.
- Disks run into bottlenecks.
- The log splitting task is overloaded, and RegionServer runs slowly.
- RegionServer (region opening) responds slowly.

In the preceding scenarios, you are advised to add the following configuration parameters to enable HMaster to complete the restoration task earlier. Otherwise, the Master will exit, causing a longer delay of the entire restoration process.

- Increase the online waiting timeout period of the namespace table to ensure that the Master has enough time to coordinate the splitting tasks of the RegionServer worker and avoid repeated tasks.

  hbase.master.namespace.init.timeout (default value: 3,600,000 ms)

- Increase the number of concurrent splitting tasks through RegionServer worker to ensure that RegionServer worker can process splitting tasks in parallel (RegionServers need more cores). Add the following parameters to *Client installation path* **/HBase/hbase/conf/hbase-site.xml**:

  hbase.regionserver.wal.max.splitters (default value: 2)

- If all restoration processes require time, increase the timeout period for initializing the monitoring thread.

  hbase.master.initializationmonitor.timeout (default value: 3,600,000 ms)

9.9.4 Why Does SocketTimeoutException Occur When a Client Queries HBase?

Question

Why does the following exception occur on the client when I use the HBase client to operate table data?

```
2015-12-15 02:41:14,054 | WARN  | [task-result-getter-2] | Lost task 2.0 in stage 58.0 (TID 3288, linux-175):
org.apache.hadoop.hbase.client.RetriesExhaustedException: Failed after attempts=36, exceptions:
Tue Dec 15 02:41:14 CST 2015, null, java.net.SocketTimeoutException: callTimeout=60000,
callDuration=60303:
row 'xxxxxx' on table 'xxxxxx' at region=xxxxxx,\x05\x1E
\x80\x00\x00\x00\x80\x00\x00\x00\x00\x00\x00\x00\x80\x00\x00\x00\x00\x00\x000\x00\x80\x00\x00\x0
0\x80\x00\x00\x00\x80\x00\x00,
1449912620868.6a6b7d0c272803d8186930a3bfdb10a9., hostname=xxxxxx,16020,1449941841479,
seqNum=5
at
org.apache.hadoop.hbase.client.RpcRetryingCallerWithReadReplicas.throwEnrichedException(RpcRetryingCall
erWithReadReplicas.java:275)
at org.apache.hadoop.hbase.client.ScannerCallableWithReplicas.call(ScannerCallableWithReplicas.java:223)
at org.apache.hadoop.hbase.client.ScannerCallableWithReplicas.call(ScannerCallableWithReplicas.java:61)
at org.apache.hadoop.hbase.client.RpcRetryingCaller.callWithoutRetries(RpcRetryingCaller.java:200)
at org.apache.hadoop.hbase.client.ClientScanner.call(ClientScanner.java:323)
```

At the same time, the following log is displayed on RegionServer:

```
2015-12-15 02:45:44,551 | WARN  | PriorityRpcServer.handler=7,queue=1,port=16020 | (responseTooSlow):
{"call":"Scan(org.apache.hadoop.hbase.protobuf.generated.ClientProtos$ScanRequest)
","starttimems":1450118730780,"responsesize":416,"method":"Scan","processingtimems":13770,"client":"10.9
1.8.175:41182","queuetimems":0,"class":"HRegionServer"} |
org.apache.hadoop.hbase.ipc.RpcServer.logResponse(RpcServer.java:2221)
2015-12-15 02:45:57,722 | WARN  | PriorityRpcServer.handler=3,queue=1,port=16020 | (responseTooSlow):
{"call":"Scan(org.apache.hadoop.hbase.protobuf.generated.ClientProtos
$ScanRequest)","starttimems":1450118746297,"responsesize":416,
"method":"Scan","processingtimems":11425,"client":"10.91.8.175:41182","queuetimems":1746,"class":"HRegi
onServer"} | org.apache.hadoop.hbase.ipc.RpcServer.logResponse(RpcServer.java:2221)
2015-12-15 02:47:21,668 | INFO  | LruBlockCacheStatsExecutor | totalSize=7.54 GB, freeSize=369.52 MB,
max=7.90 GB, blockCount=406107,
accesses=35400006, hits=16803205, hitRatio=47.47%, , cachingAccesses=31864266, cachingHits=14806045,
cachingHitsRatio=46.47%,
evictions=17654, evicted=16642283, evictedPerRun=942.69189453125 |
org.apache.hadoop.hbase.io.hfile.LruBlockCache.logStats(LruBlockCache.java:858)
2015-12-15 02:52:21,668 | INFO  | LruBlockCacheStatsExecutor | totalSize=7.51 GB, freeSize=395.34 MB,
max=7.90 GB, blockCount=403080,
accesses=35685793, hits=16933684, hitRatio=47.45%, , cachingAccesses=32150053, cachingHits=14936524,
cachingHitsRatio=46.46%,
```



```
evictions=17684, evicted=16800617, evictedPerRun=950.046142578125 |
org.apache.hadoop.hbase.io.hfile.LruBlockCache.logStats(LruBlockCache.java:858)
```

Answer

The memory allocated to RegionServer is too small and the number of Regions is too large. As a result, the memory is insufficient during the running, and the server responds slowly to the client. Modify the following memory allocation parameters in the **hbase-site.xml** configuration file of RegionServer:

Table 9-35 RegionServer memory allocation parameters

Parameter	Description	Default Value
GC_OPTS	Initial memory and maximum memory allocated to RegionServer in startup parameters.	-Xms8G -Xmx8G
hfile.block.cache.size	Percentage of the maximum heap (-Xmx setting) allocated to the block cache of HFiles or StoreFiles.	When **offheap** is disabled, the default value is **0.25**. When **offheap** is enabled, the default value is **0.1**.

9.9.5 Why "java.lang.UnsatisfiedLinkError: Permission denied" exception thrown while starting HBase shell?

Question

Why "java.lang.UnsatisfiedLinkError: Permission denied" exception thrown while starting HBase shell?

Answer

During HBase shell execution JRuby create temporary files under **java.io.tmpdir** path and default value of **java.io.tmpdir** is **/tmp**. If NOEXEC permission is set to /tmp directory then HBase shell start will fail with "java.lang.UnsatisfiedLinkError: Permission denied" exception.

So "java.io.tmpdir" must be set to a different path in HBASE_OPTS/ CLIENT_GC_OPTS if NOEXEC is set to /tmp directory.

9.9.6 When does the RegionServers listed under "Dead Region Servers" on HMaster WebUI gets cleared?

Question

When does the RegionServers listed under "Dead Region Servers" on HMaster WebUI gets cleared?

Answer

When an online RegionServer goes down abruptly, it is displayed under "Dead Region Servers" in the HMaster WebUI. When dead RegionServer restarts and reports back to HMaster successfully, the "Dead Region Servers" in the HMaster WebUI gets cleared.

The "Dead Region Servers" is also gets cleared, when the HMaster failover operation is performed successfully.

In cases when an Active HMaster hosting some regions is abruptly killed, Backup HMaster will become the new Active HMater and displays previous Active HMaster as dead RegionServer.

9.9.7 Insufficient Rights When Accessing Phoenix

Question

When a tenant accesses Phoenix, a message is displayed indicating that the tenant has insufficient rights.

Answer

You need to associate the HBase service and Yarn queues when creating a tenant.

The tenant must be granted additional rights to perform operations on Phoenix, that is, the RWX permission on the Phoenix system table.

Example:

Tenant **hbase**has been created. Log in to the HBase Shell as user**admin**and run the**scan 'hbase:acl'**command to query the role of the tenant. The role is**hbase_1450761169920** (in the format of tenant name_timestamp).

Run the following commands to grant rights to the tenant (if the Phoenix system table has not been generated, log in to the Phoenix client as user **admin** first and then grant rights on the HBase Shell):

grant '@hbase_1450761169920','RWX','SYSTEM.CATALOG'

grant '@hbase_1450761169920','RWX','SYSTEM.FUNCTION'

grant '@hbase_1450761169920','RWX','SYSTEM.SEQUENCE'

grant '@hbase_1450761169920','RWX','SYSTEM.STATS'

Create user **phoenix**and bind it with tenant**hbase**, so that tenant**hbase**can access the Phoenix client as user**phoenix**.

9.9.8 Insufficient Rights When Useing the HBase Bulkload Function

Question

When a tenant uses the HBase bulkload function, a message is displayed indicating that the tenant has insufficient rights.

Answer

You need to associate the HBase service and Yarn queues when creating a tenant.

Example:

Create user **user** and bind it with the tenant role with the same name.

User **user** must be granted additional rights to use the bulkload function.

The following **uses** user user as an example.

For details, see *Importing Data in Batches*. The following describes the differences:

1. Create the data file directory in **/tmp** and run the following commands:

   hdfs dfs -mkdir /tmp/datadirImport

   hdfs dfs -put data.txt /tmp/datadirImport

2. Use the **/tmp** directory of HDFS to generate HFile:

   hbase com.huawei.hadoop.hbase.tools.bulkload.ImportData -Dimport.skip.bad.lines=true -Dimport.separator=',' -Dimport.bad.lines.output=/tmp/badline -Dimport.hfile.output=/tmp/hfile configuration.xml ImportTable /tmp/datadirImport

3. Use the **/tmp** directory of HDFS to import HFile:

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /tmp/hfile ImportTable

9.9.9 How Do I Fix Region Overlapping?

Question

When the hbck tool is used to check the region status, if the log contains **ERROR: (regions region1 and region2) There is an overlap in the region chain** or **ERROR: (region region1) Multiple regions have the same startkey: xxx**, overlapping exists in some regions. How do I solve this problem?

Answer

To rectify the fault, perform the following steps:

Step 1 Run the **hbase hbck -j** ${CLIENT_HOME}**/HBase/hbase/tools/hbase-hbck2-1.1.0-h0.cbu.mrs.*.jar fixInconsistencies** *tableName* command to restore the table that contains overlapping.

Step 2 Run the **hbase hbck -j** ${CLIENT_HOME}**/HBase/hbase/tools/hbase-hbck2-1.1.0-h0.cbu.mrs.*.jar listInconsistencies -run** *tableName* command to check whether overlapping exists in the restored table.

- If overlapping does not exist, go to **Step 3**.
- If overlapping exists, go to **Step 1**.

Step 3 Log in to FusionInsight Manager and choose **Cluster** > *Name of the desired cluster* > **Services** > **HBase** > **More** > **Perform HMaster Switchover** to complete the HMaster active/standby switchover.

Step 4 Run the **hbase hbck -j** ${CLIENT_HOME}**/HBase/hbase/tools/hbase-hbck2-1.1.0-h0.cbu.mrs.*.jar listInconsistencies -run** *tableName* command to check whether overlapping exists in the restored table.

- If overlapping does not exist, no further action is required.
- If overlapping still exists, start from **Step 1** to perform the recovery again.

----End

9.9.10 Restrictions on using the Phoenix BulkLoad Tool

Question

When the indexed field data is updated, if a batch of data exists in the user table, the BulkLoad tool cannot update the global and partial mutable indexes.

Answer

Problem Analysis

1. Create a table.

```
CREATE TABLE TEST_TABLE(
DATE varchar not null,
NUM integer not null,
SEQ_NUM integer not null,
ACCOUNT1 varchar not null,
ACCOUNTDES varchar,
FLAG varchar,
SALL double,
CONSTRAINT PK PRIMARY KEY (DATE,NUM,SEQ_NUM,ACCOUNT1)
);
```

2. Create a global index.

   CREATE INDEX TEST_TABLE_INDEX ON TEST_TABLE(ACCOUNT1,DATE,NUM,ACCOUNTDES,SEQ_NUM);

3. Insert data.

   UPSERT INTO TEST_TABLE (DATE,NUM,SEQ_NUM,ACCOUNT1,ACCOUNTDES,FLAG,SALL) values ('20201001',30201001,13,'367392332','sffa1','','');

4. Execute the BulkLoad task to update data.

   hbase org.apache.phoenix.mapreduce.CsvBulkLoadTool -t TEST_TABLE -i /tmp/test.csv, where the content of **test.csv** is as follows:

20201001	30201001	13	367392332	sffa888	1231243	23

5. Symptom: The existing index data cannot be directly updated. As a result, two pieces of index data exist.

```
+------------+-----------+-----------+---------------+----------------+
| :ACCOUNT1  |   :DATE   |   :NUM    | 0:ACCOUNTDES  | :SEQ_NUM  |
+------------+-----------+-----------+---------------+----------------+
| 367392332  | 20201001  | 30201001  | sffa1         | 13                |
| 367392332  | 20201001  | 30201001  | sffa888       | 13                |
+------------+-----------+-----------+---------------+----------------+
```

Solution

Step 1 Delete the old index table.

DROP INDEX TEST_TABLE_INDEX ON TEST_TABLE;

Step 2 Create an index table in asynchronous mode.

CREATE INDEX TEST_TABLE_INDEX ON TEST_TABLE(ACCOUNT1,DATE,NUM,ACCOUNTDES,SEQ_NUM) ASYNC;

Step 3 Recreate a index.

hbase org.apache.phoenix.mapreduce.index.IndexTool --data-table TEST_TABLE --index-table TEST_TABLE_INDEX --output-path /user/test_table

----End

9.9.11 Why a Message Is Displayed Indicating that the Permission is Insufficient When CTBase Connects to the Ranger Plug-ins?

Question

When CTBase accesses the HBase service with the Ranger plug-ins enabled and you are creating a cluster table, a message is displayed indicating that the permission is insufficient.

The error information is as follows:

```
ERROR: Create ClusterTable failed. Error: org.apache.hadoop.hbase.security.AccessDeniedException:
Insufficient permissions for user 'ctbase2@HADOOP.COM' (action=create)
at org.apache.ranger.authorization.hbase.AuthorizationSession.publishResults(AuthorizationSession.java:278)
at
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.authorizeAccess(RangerAuthorizatio
nCoprocessor.java:654)
at
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.requirePermission(RangerAuthorizati
onCoprocessor.java:772)
at
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.preCreateTable(RangerAuthorization
Coprocessor.java:943)
at
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.preCreateTable(RangerAuthorization
Coprocessor.java:428)
at org.apache.hadoop.hbase.master.MasterCoprocessorHost$12.call(MasterCoprocessorHost.java:351)
at org.apache.hadoop.hbase.master.MasterCoprocessorHost$12.call(MasterCoprocessorHost.java:348)
at org.apache.hadoop.hbase.coprocessor.CoprocessorHost
$ObserverOperationWithoutResult.callObserver(CoprocessorHost.java:581)
at org.apache.hadoop.hbase.coprocessor.CoprocessorHost.execOperation(CoprocessorHost.java:655)
at
org.apache.hadoop.hbase.master.MasterCoprocessorHost.preCreateTable(MasterCoprocessorHost.java:348)
```



```
at org.apache.hadoop.hbase.master.HMaster$5.run(HMaster.java:2192)
at
org.apache.hadoop.hbase.master.procedure.MasterProcedureUtil.submitProcedure(MasterProcedureUtil.java:1
34)
at org.apache.hadoop.hbase.master.HMaster.createTable(HMaster.java:2189)
at org.apache.hadoop.hbase.master.MasterRpcServices.createTable(MasterRpcServices.java:711)
at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService
$2.callBlockingMethod(MasterProtos.java)
at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:458)
at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:133)
at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:338)
at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:318)
```

Answer

Check whether the current account has sufficient permissions.

The CTBase user needs to configure permission policies on the Ranger page and grant the READ, WRITE, CREATE, ADMIN, and EXECUTE permissions to the CTBase metadata table _**ctmeta**_, cluster table, and index table.

For details about how to configure permissions on the Ranger page, see **Adding a Ranger Access Permission Policy for HBase**.

9.9.12 Introduction to HBase Global Secondary Index APIs

APIs that use global indexes are in the org.apache.hadoop.hbase.hindex.global.GlobalIndexAdmin class. Related APIs are described as follows:

Operation	API	Description
Adding an index	addIndices()	Add an index to a table without data. Calling this API will add the specified index to a table but skips index data generation. This API is used to add indexes in batches to a table that contains a large amount of pre-existing user data and use the GlobalTableIndexer tool to build index data.
	addIndicesWithDa-ta()	Add an index to a table with data. This API is used to add the specified index to the table and create index data for the existing user data. Alternatively, the API can be called to generate an index and then generate index data when the user data is being stored. This API is not recommended when a data table contains a large amount of data.
Deleting a collection	dropIndices()	If only the index is deleted, both the index metadata and index data are deleted. After this operation, the index cannot be used for scan or filter operations.

Operation	API	Description
Modifying the Index Status	alterGlobalIndicesUnusable()	Disables a specified index so that it cannot be used for scan or filter operations.
	alterGlobalIndicesActive()	Enables the index specified by the user so that it can be used for the scan/filter operation.
	alterGlobalIndicesInactive()	This API is used to disable a specified index and cancel index data generation. It cannot be used for scan or filter operations. It is usually used in the index repair process.
Viewing the created index	listIndices()	It can be used to list all indexes in a given table.

9.9.13 How Do I Disable HDFS Hedged Read on HBase?

Symptom

For MRS 3.3.1 and later versions, HDFS hedged read is enabled by default to reduce read latency and adapt to network changes. For details about this function, see **Configuring HDFS Hedged Read**. **Table 9-36** describes related parameters.

Table 9-36 Parameters of HDFS hedged read

Parameter	Description	Default Value	Value Range
dfs.client.hedged.read.threshold.millis	The number of milliseconds the HDFS client waits for the first byte of the first data block before deciding whether to start a hedged read	250	Greater than or equal to 1
dfs.client.hedged.read.threadpool.size	Size of the hedged read thread pool. If this parameter is set to a value greater than 0, multiple read channels are enabled.	200	Greater than or equal to 0

HDFS hedged read may cause performance deterioration when the disk I/O is high. You need to disable this function on HBase by referring to **Procedure**.

Procedure

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Service** > **HBase** > **Configurations** > **All Configurations**. The **All Configurations** page is displayed.

Step 3 Search for **dfs.client.hedged.read.threadpool.size** and change its value to **0**.

Step 4 Click **Save**.

Step 5 Click **Instances**, select all RegionServer instances, and choose **More** > **Instance Rolling Restart** to apply the changes.

----End

9.10 HBase Troubleshooting

9.10.1 Why Does a Client Keep Failing to Connect to a Server for a Long Time?

Question

A HBase server is faulty and cannot provide services. In this case, when a table operation is performed on the HBase client, why is the operation suspended and no response is received for a long time?

Answer

Problem Analysis

When the HBase server malfunctions, the table operation request from the HBase client is tried for several times and times out. The default timeout value is **Integer.MAX_VALUE (2147483647 ms)**. The table operation request is retired constantly during such a long period of time and is suspended at last.

Solution

The HBase client provides two configuration items to configure the retry and timeout of the client. **Table 9-37** describes them.

Set the following parameters in the ***Client installation path*/HBase/hbase/conf/hbase-site.xml** configuration file:

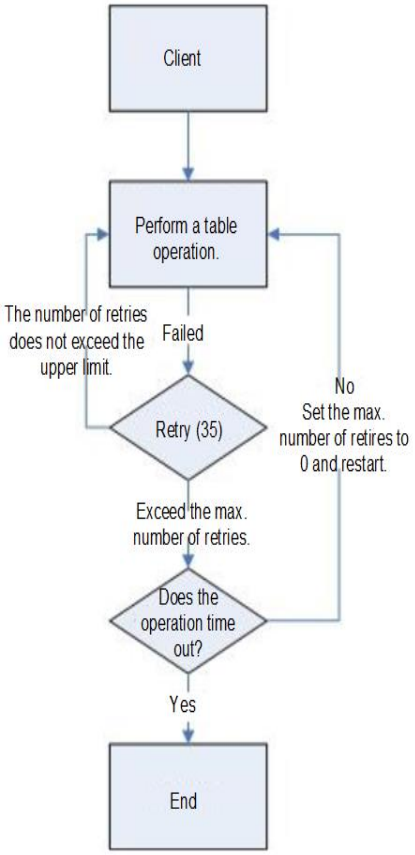
Table 9-37 Configuration parameters of retry and timeout

Parameter	Description	Default Value
hbase.client.operation.timeout	Client operation timeout period You need to manually add the information to the configuration file.	2147483647 ms

Parameter	Description	Default Value
hbase.client.retries.number	Maximum retry times supported by all retryable operations.	35

Figure 9-6 describes the working principles of retry and timeout.

Figure 9-6 Process for HBase client operation retry timeout



The process indicates that a suspension occurs if the preceding parameters are not configured based on site requirements. It is recommended that a proper timeout period be set based on scenarios. If the operation takes a long time, set a long timeout period. If the operation takes a shot time, set a short timeout period. The number of retries can be set to **(hbase.client.retries.number)*60*1000(ms)**. The timeout period can be slightly greater than **hbase.client.operation.timeout**.

9.10.2 Why May a Table Creation Exception Occur When HBase Deletes or Creates the Same Table Consecutively?

Question

When HBase consecutively deletes and creates the same table, why may a table creation exception occur?

Answer

Execution process: Disable Table > Drop Table > Create Table > Disable Table > Drop Table > And more

1. When a table is disabled, HMaster sends an RPC request to RegionServer, and RegionServer brings the region offline. When the time required for closing a region on RegionServer exceeds the timeout period for HBase HMaster to wait for the region to enter the RIT state, HMaster considers that the region is offline by default. Actually, the region may be in the flush memstore phase.
2. After an RPC request is sent to close a region, HMaster checks whether all regions in the table are offline. If the closure times out, HMaster considers that the regions are offline and returns a message indicating that the regions are successfully closed.
3. After the closure is successful, the data directory corresponding to the HBase table is deleted.
4. After the table is deleted, the data directory is recreated by the region that is still in the flush memstore phase.
5. When the table is created again, the **temp** directory is copied to the HBase data directory. However, the HBase data directory is not empty. As a result, when the HDFS rename API is called, the data directory changes to the last layer of the **temp** directory and is appended to the HBase data directory, for example, **$rootDir/data/$nameSpace/$tableName/$tableName**. In this case, the table fails to be created.

Troubleshooting Method

When this problem occurs, check whether the HBase data directory corresponding to the table exists. If it exists, rename the directory.

The HBase data directory consists of **$rootDir/data/$nameSpace/$tableName**, for example, **hdfs://hacluster/hbase/data/default/TestTable**. **$rootDir** is the HBase root directory, which can be obtained by configuring **hbase.rootdir.perms** in **hbase-site.xml**. The **data** directory is a fixed directory of HBase. **$nameSpace** indicates the nameSpace name. **$tableName** indicates the table name.

9.10.3 Why Other Services Become Unstable If HBase Sets up A Large Number of Connections over the Network Port?

Question

Why other services become unstable if HBase sets up a large number of connections over the network port?

Answer

When the OS command ***lsof*** or ***netstat*** is run, it is found that many TCP connections are in the CLOSE_WAIT state and the owner of the connections is HBase RegionServer. This can cause exhaustion of network ports or limit exceeding of HDFS connections, resulting in instability of other services. The HBase CLOSE_WAIT phenomenon is the HBase mechanism.

The reason why HBase CLOSE_WAIT occurs is as follows: HBase data is stored in the HDFS as HFile, which can be called StoreFiles. HBase functions as the client of the HDFS. When HBase creates a StoreFile or starts loading a StoreFile, it creates an HDFS connection. When the StoreFile is created or loaded successfully, the HDFS considers that the task is completed and transfers the connection close permission to HBase. However, HBase may choose not to close the connection to ensure real-time response; that is, HBase may maintain the connection so that it can quickly access the corresponding data file upon request. In this case, the connection is in the CLOSE_WAIT, which indicates that the connection needs to be closed by the client.

When a StoreFile will be created: HBase executes the Flush operation.

When Flush is executed: The data written by HBase is first stored in memstore. The Flush operation is performed only when the usage of memstore reaches the threshold or the ***flush*** command is run to write data into the HDFS.

To resolve the issue, use either of the following methods:

Because of the HBase connection mechanism, the number of StoreFiles must be restricted to reduce the occupation of HBase ports. This can be achieved by triggering HBase's the compaction action, that is, HBase file merging.

Method 1: On HBase shell client, run ***major_compact***.

Method 2: Compile HBase client code to invoke the compact method of the HBaseAdmin class to trigger HBase's compaction action.

If the HBase port occupation issue cannot be resolved through compact, it indicates that the HBase usage has reached the bottleneck. In such a case, you are advised to perform the following:

- Check whether the initial number of Regions configured in the table is appropriate.
- Check whether useless data exists.

If useless data exists, delete the data to reduce the number of storage files for the HBase. If the preceding conditions are not met, then you need to consider a capacity expansion.

9.10.4 Why Does the HBase BulkLoad Task Consisting of 210,000 Map Tasks and 10,000 Reduce Tasks Fail?

Question

The HBase bulkLoad task (a single table contains 26 TB data) has 210,000 maps and 10,000 reduce tasks, and the task fails.

Answer

ZooKeeper I/O bottleneck observation methods:

1. On the monitoring page of Manager, check whether the number of ZooKeeper requests on a single node exceeds the upper limit.
2. View ZooKeeper and HBase logs to check whether a large number of I/O Exception Timeout or SocketTimeout Exception exceptions occur.

Optimization suggestions:

1. Change the number of ZooKeeper instances to 5 or more. You are advised to set **peerType** to **observer** to increase the number of observers.
2. Control the number of concurrent maps of a single task or reduce the memory for running tasks on each node to lighten the node load.
3. Upgrade ZooKeeper data disks, such as SSDs.

9.10.5 Why Modified and Deleted Data Can Still Be Queried by Using the Scan Command?

Question

Why modified and deleted data can still be queried by using the **scan** command?

```
scan '<table_name>',{FILTER=>"SingleColumnValueFilter('<column_family>','column',=,'binary:<value>')"}
```

Answer

Because of the scalability of HBase, all values specific to the versions in the queried column are all matched by default, even if the values have been modified or deleted. For a row where column matching has failed (that is, the column does not exist in the row), the HBase also queries the row.

If you want to query only the new values and rows where column matching is successful, you can use the following statement:

```
scan '<table_name>',
{FILTER=>"SingleColumnValueFilter('<column_family>','column',=,'binary:<value>',true,true)"}
```

This command can filter all rows where column query has failed. It queries only the latest values of the current data in the table; that is, it does not query the values before modification or the deleted values.

NOTE

The related parameters of **SingleColumnValueFilter** are described as follows:

SingleColumnValueFilter(final byte[] family, final byte[] qualifier, final CompareOp compareOp, ByteArrayComparable comparator, final boolean filterIfMissing, final boolean latestVersionOnly)

Parameter description:

- family: family of the column to be queried.
- qualifier: column to be queried.
- compareOp: comparison operation, such as = and >.
- comparator: target value to be queried.
- filterIfMissing: whether a row is filtered out if the queried column does not exist. The default value is false.
- latestVersionOnly: whether values of the latest version are queried. The default value is false.

9.10.6 What Should I Do If I Fail to Create Tables Due to the FAILED_OPEN State of Regions?

Question

What should I do if I fail to create tables due to the FAILED_OPEN state of Regions?

Answer

If a network, HDFS, or Active HMaster fault occurs during the creation of tables, some Regions may fail to go online and therefore enter the FAILED_OPEN state. In this case, tables fail to be created.

The tables that fail to be created due to the preceding mentioned issue cannot be repaired. To solve this problem, perform the following operations to delete and re-create the tables:

1. Run the following command on the cluster client to repair the state of the tables:

   hbase hbck -j ${CLIENT_HOME}**/HBase/hbase/tools/hbase-hbck2-1.1.0-h0.cbu.mrs.*.jar setTableState** <table_name> **ENABLED**

2. Enter the HBase shell and run the following commands to delete the tables that fail to be created:

   disable *'<table_name>'*

   drop *'<table_name>'*

3. Create the tables using the recreation command.

9.10.7 How Do I Delete Residual Table Names in the table-lock Directory of ZooKeeper?

Question

In security mode, names of tables that failed to be created are unnecessarily retained in the table-lock node (default directory is /hbase/table-lock) of ZooKeeper. How do I delete these residual table names?

Answer

Perform the following steps:

1. On the client, run the kinit command as the hbase user to obtain a security certificate.
2. Run the ***hbase zkcli*** command to launch the ZooKeeper Command Line Interface (zkCLI).
3. Run the ***ls /hbase/table*** command on the zkCLI to check whether the table name of the table that fails to be created exists.
   - If the table name exists, no further operation is required.
   - If the table name does not exist, run ***ls /hbase/table-lock***to check whether the table name of the table fail to be created exist. If the table name exists, run the***delete /hbase/table-lock/<table>***command to delete the table name. In the***delete /hbase/table-lock/<table>***command,***<table>*** indicates the residual table name.

9.10.8 Why Does HBase Become Faulty When I Set a Quota for the Directory Used by HBase in HDFS?

Question

Why does HBase become faulty when I set quota for the directory used by HBase in HDFS?

Answer

The flush operation of a table is to write memstore data to HDFS.

If the HDFS directory does not have sufficient disk space quota, the flush operation will fail and the region server will stop.

```
Caused by: org.apache.hadoop.hdfs.protocol.DSQuotaExceededException: The DiskSpace quota of /hbase/data/<namespace>/<tableName> is exceeded: quota = 1024 B = 1 KB but diskspace consumed = 402655638 B = 384.00 MB
?at org.apache.hadoop.hdfs.server.namenode.DirectoryWithQuotaFeature.verifyStoragespaceQuota(DirectoryWithQuotaFeature.java:211)
?at org.apache.hadoop.hdfs.server.namenode.DirectoryWithQuotaFeature.verifyQuota(DirectoryWithQuotaFeature.java:239)
?at org.apache.hadoop.hdfs.server.namenode.FSDirectory.verifyQuota(FSDirectory.java:882)
?at org.apache.hadoop.hdfs.server.namenode.FSDirectory.updateCount(FSDirectory.java:711)
?at org.apache.hadoop.hdfs.server.namenode.FSDirectory.updateCount(FSDirectory.java:670)
?at org.apache.hadoop.hdfs.server.namenode.FSDirectory.addBlock(FSDirectory.java:495)
```


In the preceding exception, the disk space quota of the **/hbase/data/<namespace>/<tableName>** table is 1 KB, but the memstore data is 384.00 MB. Therefore, the flush operation fails and the region server stops.

When the region server is terminated, HMaster replays the WAL file of the terminated region server to restore data. The disk space quota is limited. As a result, the replay operation of the WAL file fails, and the HMaster process exits unexpectedly.

```
2016-07-28 19:11:40,352 | FATAL | MASTER_SERVER_OPERATIONS-10-91-9-131:16000-0 | Caught throwable while processing event M_SERVER_SHUTDOWN |
org.apache.hadoop.hbase.master.HMaster.abort(HMaster.java:2474)
java.io.IOException: failed log splitting for 10-91-9-131,16020,1469689987884, will retry
?at org.apache.hadoop.hbase.master.handler.ServerShutdownHandler.resubmit(ServerShutdownHandler.java:365)
?at org.apache.hadoop.hbase.master.handler.ServerShutdownHandler.process(ServerShutdownHandler.java:220)
?at org.apache.hadoop.hbase.executor.EventHandler.run(EventHandler.java:129)
?at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
?at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
?at java.lang.Thread.run(Thread.java:745)
Caused by: java.io.IOException: error or interrupted while splitting logs in [hdfs://hacluster/hbase/WALs/<RS-Hostname>,<RS-Port>,<startcode>-splitting] Task = installed = 6 done = 3 error = 3
?at org.apache.hadoop.hbase.master.SplitLogManager.splitLogDistributed(SplitLogManager.java:290)
?at org.apache.hadoop.hbase.master.MasterFileSystem.splitLog(MasterFileSystem.java:402)
?at org.apache.hadoop.hbase.master.MasterFileSystem.splitLog(MasterFileSystem.java:375)
```

Therefore, you cannot set the quota value for the HBase directory in HDFS. If the exception occurs, perform the following operations:

Step 1 Run the **kinit** *Username* command on the client to enable the HBase user to obtain security authentication.

Step 2 Run the **hdfs dfs -count -q** */hbase/data/<namespace>/<tableName>* command to check the allocated disk space quota.

Step 3 Run the following command to cancel the quota limit and restore HBase:

hdfs dfsadmin -clrSpaceQuota */hbase/data/<namespace>/<tableName>*

----End

9.10.9 HMaster Fails to Be Started After the OfflineMetaRepair Tool Is Used to Rebuild Metadata

Question

Why HMaster times out while waiting for namespace table to be assigned after rebuilding meta using OfflineMetaRepair tool and startups failed?

HMaster abort with following FATAL message,

```
2017-06-15 15:11:07,582 FATAL [Hostname:16000.activeMasterManager] master.HMaster: Unhandled exception. Starting shutdown.
java.io.IOException: Timedout 120000ms waiting for namespace table to be assigned
    at org.apache.hadoop.hbase.master.TableNamespaceManager.start(TableNamespaceManager.java:98)
    at org.apache.hadoop.hbase.master.HMaster.initNamespace(HMaster.java:1054)
    at org.apache.hadoop.hbase.master.HMaster.finishActiveMasterInitialization(HMaster.java:848)
    at org.apache.hadoop.hbase.master.HMaster.access$600(HMaster.java:199)
    at org.apache.hadoop.hbase.master.HMaster$2.run(HMaster.java:1871)
    at java.lang.Thread.run(Thread.java:745)
```


Answer

When meta is rebuilt by OfflineMetaRepair tool then HMaster wait for all region server's WAL split during start up to avoid the data inconsistency problem. HMaster trigger user regions assignment once WAL split completes. So when the cluster is in the unusual scenario, there are chances WAL splitting may take long time which depends on multiple factors like too many WALs, slow I/O, region servers are not stable etc.

HMaster should be able to finish all region server WAL splitting successfully. Perform the following steps.

1. Make sure cluster is stable, no other problem exist. If any problem occurs, please correct them first.
2. Configure a large value to **hbase.master.initializationmonitor.timeout**parameters, default value is**3600000** milliseconds.
3. Restart HBase service.

9.10.10 Why Messages Containing FileNotFoundException Frequently Displayed in the HMaster Logs?

Question

Why messages containing FileNotFoundException and no lease are frequently displayed in the HMaster logs during the WAL splitting process?

```
2017-06-10 09:50:27,586 | ERROR | split-log-closeStream-2 | Couldn't close log at hdfs://hacluster/hbase/
data/default/largeT1/2b48346d087275fe751fc049334fda93/recovered.edits/0000000000000000000.temp |
org.apache.hadoop.hbase.wal.WALSplitter$LogRecoveredEditsOutputSink$2.call(WALSplitter.java:1330)
java.io.FileNotFoundException: No lease on /hbase/data/default/
largeT1/2b48346d087275fe751fc049334fda93/recovered.edits/0000000000000000000.temp (inode
1092653): File does not exist. [Lease.  Holder: DFSClient_NONMAPREDUCE_1202985678_1, pendingcreates:
1936]
?at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkLease(FSNamesystem.java:3432)
?at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.analyzeFileState(FSNamesystem.java:3223)
?at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getNewBlockTargets(FSNamesystem.java:3057)
?at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getAdditionalBlock(FSNamesystem.java:3011)
?at org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.addBlock(NameNodeRpcServer.java:842)
?at
org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTranslatorPB.addBlock(ClientNameno
deProtocolServerSideTranslatorPB.java:526)
?at org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$ClientNamenodeProtocol
$2.callBlockingMethod(ClientNamenodeProtocolProtos.java)
?at org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.call(ProtobufRpcEngine.java:616)
?at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:973)
?at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2260)
?at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2256)
?at java.security.AccessController.doPrivileged(Native Method)
?at javax.security.auth.Subject.doAs(Subject.java:422)
?at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1769)
?at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2254)

?at sun.reflect.GeneratedConstructorAccessor40.newInstance(Unknown Source)
?at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
?at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
?at org.apache.hadoop.ipc.RemoteException.instantiateException(RemoteException.java:106)
?at org.apache.hadoop.ipc.RemoteException.unwrapRemoteException(RemoteException.java:73)
?at org.apache.hadoop.hdfs.DataStreamer.locateFollowingBlock(DataStreamer.java:1842)
?at org.apache.hadoop.hdfs.DataStreamer.nextBlockOutputStream(DataStreamer.java:1639)
?at org.apache.hadoop.hdfs.DataStreamer.run(DataStreamer.java:665)
```


Answer

During the WAL splitting process, the WAL splitting timeout period is specified by the **hbase.splitlog.manager.timeout** parameter. If the WAL splitting process fails to complete within the timeout period, the task is submitted again. Multiple WAL splitting tasks may be submitted during a specified period. If the **temp** file is deleted when one WAL splitting task completes, other tasks cannot find the file and the FileNotFoudException exception is reported. To avoid the problem, perform the following modifications:

The default value of **hbase.splitlog.manager.timeout** is 600,000 ms. The cluster specification is that each RegionServer has 2,000 to 3,000 regions. When the cluster is normal (HBase is normal and HDFS does not have a large number of read and write operations), you are advised to adjust this parameter based on the cluster specifications. If the actual specifications (the actual average number of regions on each RegionServer) are greater than the default specifications (the default average number of regions on each RegionServer, that is, 2,000), the adjustment solution is (actual specifications/default specifications) x Default time.

Set the **splitlog** parameter in the **hbase-site.xml** file on the server. **Table 9-38** describes the parameter.

Table 9-38 Description of the **splitlog** parameter

Parameter	Description	Default Value
hbase.splitlog.manager.timeout	Timeout period for receiving worker response by the distributed SplitLog management program.	600000

9.10.11 Why Does the ImportTsv Tool Display "Permission denied"

Question

When the same Linux user (for example, user **omm**) as and a different Kerberos user (for example, user **admin**) from the Region Server are used, why does the ImportTsv tool fail to be executed and the error message "Permission denied" is displayed?

```
Exception in thread "main" org.apache.hadoop.security.AccessControlException: Permission denied:
user=admin, access=WRITE, inode="/user/omm-bulkload/hbase-staging/
partitions_cab16de5-87c2-4153-9cca-a6f4ed4278a6":hbase:hadoop:drwx--x--x
        at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:342)
        at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:315)
        at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:23
1)
        at
com.xxx.hadoop.adapter.hdfs.plugin.HWAccessControlEnforce.checkPermission(HWAccessControlEnforce.java:
69)
        at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:19
0)
        at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermission(FSDirectory.java:1789)
```



```
        at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermission(FSDirectory.java:1773)
        at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkAncestorAccess(FSDirectory.java:1756)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.startFileInternal(FSNamesystem.java:2490)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.startFileInt(FSNamesystem.java:2425)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.startFile(FSNamesystem.java:2308)
        at
org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.create(NameNodeRpcServer.java:745)
        at
org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTranslatorPB.create(ClientNamenodeP
rotocolServerSideTranslatorPB.java:434)
        at org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$ClientNamenodeProtocol
$2.callBlockingMethod(ClientNamenodeProtocolProtos.java)
        at org.apache.hadoop.ipc.ProtobufRpcEngine$Server
$ProtoBufRpcInvoker.call(ProtobufRpcEngine.java:616)
        at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:973)
        at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2260)
        at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2256)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:422)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1781)
        at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2254)
```

Answer

The ImportTsv tool creates a partition file in the HBase temporary directory specified by **hbase.fs.tmp.dir** in the *Client installation path* **/HBase/hbase/conf/hbase-site.xml file.** Therefore, the client (Kerberos user) must have the **rwx** permission on the specified temporary directory to perform the ImportTsv operation. The default value of **hbase.fs.tmp.dir** is **/user/${user.name}/hbase-staging** (for example, **/user/omm/hbase-staging**). **$ {user.name}** indicates the OS username (user **omm**). The client (Kerberos user, for example, user **admin**) does not have the **rwx** permission on the directory.

To solve the preceding problem, perform the following steps:

1. On the client, set **hbase.fs.tmp.dir** to the directory of the current Kerberos user (for example, **/user/admin/hbase-staging**), or provide the **rwx** permission required by the configured directory for the client (Kerberos user).
2. Perform the ImportTsv operation again.

9.10.12 Why Are Different Query Results Returned After I Use Same Query Criteria to Query Data Successfully Imported by HBase bulkload?

Question

If the data to be imported by HBase bulkload has identical rowkeys, the data import is successful but identical query criteria produce different query results.

Answer

Data with an identical rowkey is loaded into HBase in the order in which data is read. The data with the latest timestamp is considered to be the latest data. By default, data is not queried by timestamp. Therefore, if you query for data with an identical rowkey, only the latest data is returned.

While data is being loaded by bulkload, the memory processes the data into HFiles quickly, leading to the possibility that data with an identical rowkey has a

same timestamp. In this case, identical query criteria may produce different query results.

To avoid this problem, ensure that the same data file does not contain identical rowkeys while you are creating tables or loading data.

9.10.13 HBase Fails to Recover a Task

Question

The system automatically rolls back data after an HBase recovery task fails. If "Rollback recovery failed" is displayed, the rollback fails. After the rollback fails, data stops being processed and the junk data may be generated. How can I resolve this problem?

Answer

You need to manually clear the junk data before performing the backup or recovery task next time.

Step 1 Install the cluster client in **/opt/client**.

Step 2 Run **source /opt/client/bigdata_env** as the client installation user to configure environment variables.

Step 3 Run the **kinit admin** command.

Step 4 Run **zkCli.sh -server** *business IP address of ZooKeeper*:**2181** to connect to the ZooKeeper.

Step 5 Run **deleteall /recovering** to delete the junk data. Run **quit** to disconnect ZooKeeper.

NOTE

Running this command will cause data loss. Exercise caution.

Step 6 Run **hdfs dfs -rm -f -r /user/hbase/backup** to delete temporary data.

Step 7 Log in to FusionInsight Manager and choose **O&M**. In the navigation pane on the left, choose **Backup and Restoration** > **Restoration Management**. In the task list, locate the row that contains the target task and click **View History** in the **Operation** column. In the displayed dialog box, click ⌄ before a specified execution record to view the snapshot name.

Snapshot [*snapshot name*] is created successfully before recovery.

Step 8 Switch to the client, run **hbase shell**, and then **delete_all_snapshot** '*snapshot name*.*' to delete the temporary snapshot.

----End

9.10.14 Why Does RegionServer Fail to Be Started When GC Parameters Xms and Xmx of HBase RegionServer Are Set to 31 GB?

Question

Check the **hbase-omm-*.out** log of the node where RegionServer fails to be started. It is found that the log contains **An error report file with more information is saved as: /tmp/hs_err_pid*.log**. Check the **/tmp/hs_err_pid*.log** file. It is found that the log contains **#Internal Error (vtableStubs_aarch64.cpp:213), pid=9456, tid=0x0000ffff97fdd200 and #guarantee(__ pc() <= s->code_end()) failed: overflowed buffer**, indicating that the problem is caused by JDK. How do I solve this problem?

Answer

To rectify the fault, perform the following steps:

Step 1 Run the **su - omm** command on a node where RegionServer fails to be started to switch to user **omm**.

Step 2 Run the **java -XX:+PrintFlagsFinal -version |grep HeapBase** command as user **omm**. Information similar to the following is displayed:

```
uintx HeapBaseMinAddress = 2147483648 {pd product}
```

Step 3 Change the values of **-Xms** and **-Xmx** in **GC_OPTS** to values that are not between **32G-HeapBaseMinAddress** and **32G**, excluding the values of **32G** and **32G-HeapBaseMinAddress**.

Step 4 Log in to FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services** > **HBase** > **Instance**, select the failed instance, and choose **More** > **Restart Instance** to restart the failed instance.

----End

9.10.15 Why Does the LoadIncrementalHFiles Tool Fail to Be Executed and "Permission denied" Is Displayed?

Question

Why does the LoadIncrementalHFiles tool fail to be executed and "Permission denied" is displayed when a Linux user is manually created in a normal cluster and DataNode in the cluster is used to import data in batches?

```
2020-09-20 14:53:53,808 WARN  [main] shortcircuit.DomainSocketFactory: error creating DomainSocket
java.net.ConnectException: connect(2) error: Permission denied when trying to connect to '/var/run/
FusionInsight-HDFS/dn_socket'
    at org.apache.hadoop.net.unix.DomainSocket.connect0(Native Method)
    at org.apache.hadoop.net.unix.DomainSocket.connect(DomainSocket.java:256)
    at org.apache.hadoop.hdfs.shortcircuit.DomainSocketFactory.createSocket(DomainSocketFactory.java:168)
    at org.apache.hadoop.hdfs.client.impl.BlockReaderFactory.nextDomainPeer(BlockReaderFactory.java:804)
    at
org.apache.hadoop.hdfs.client.impl.BlockReaderFactory.createShortCircuitReplicaInfo(BlockReaderFactory.java
:526)
    at org.apache.hadoop.hdfs.shortcircuit.ShortCircuitCache.create(ShortCircuitCache.java:785)
```



```
    at org.apache.hadoop.hdfs.shortcircuit.ShortCircuitCache.fetchOrCreate(ShortCircuitCache.java:722)
    at
org.apache.hadoop.hdfs.client.impl.BlockReaderFactory.getBlockReaderLocal(BlockReaderFactory.java:483)
    at org.apache.hadoop.hdfs.client.impl.BlockReaderFactory.build(BlockReaderFactory.java:360)
    at org.apache.hadoop.hdfs.DFSInputStream.getBlockReader(DFSInputStream.java:663)
    at org.apache.hadoop.hdfs.DFSInputStream.blockSeekTo(DFSInputStream.java:594)
    at org.apache.hadoop.hdfs.DFSInputStream.readWithStrategy(DFSInputStream.java:776)
    at org.apache.hadoop.hdfs.DFSInputStream.read(DFSInputStream.java:845)
    at java.io.DataInputStream.readFully(DataInputStream.java:195)
    at org.apache.hadoop.hbase.io.hfile.FixedFileTrailer.readFromStream(FixedFileTrailer.java:401)
    at org.apache.hadoop.hbase.io.hfile.HFile.isHFileFormat(HFile.java:651)
    at org.apache.hadoop.hbase.io.hfile.HFile.isHFileFormat(HFile.java:634)
    at org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.visitBulkHFiles(LoadIncrementalHFiles.java:1090)
    at
org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.discoverLoadQueue(LoadIncrementalHFiles.java:1006)
    at
org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.prepareHFileQueue(LoadIncrementalHFiles.java:257)
    at org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.doBulkLoad(LoadIncrementalHFiles.java:364)
    at org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.run(LoadIncrementalHFiles.java:1263)
    at org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.run(LoadIncrementalHFiles.java:1276)
    at org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.run(LoadIncrementalHFiles.java:1311)
    at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
    at org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.main(LoadIncrementalHFiles.java:1333)
```

Answer

If the client that the LoadIncrementalHFiles tool depends on is installed in the cluster and is on the same node as DataNode, HDFS creates short-circuit read during the execution of the tool to improve performance. The short-circuit read depends on the **/var/run/FusionInsight-HDFS** directory (**dfs.domain.socket.path**). The default permission on this directory is **750**. This user does not have the permission to operate the directory.

To solve the preceding problem, perform the following operations:

Method 1: Create a user (recommended).

Step 1 Create a user on Manager. By default, the user group contains the **ficommon** group.

```
[root@xxx-xxx-xxx-xxx ~]# id test
uid=20038(test) gid=9998(ficommon) groups=9998(ficommon)
```

Step 2 Import data again.

----End

Method 2: Change the owner group of the current user.

Step 1 Add the user to the **ficommon** group.

```
[root@xxx-xxx-xxx-xxx ~]# usermod -a -G ficommon test
[root@xxx-xxx-xxx-xxx ~]# id test
uid=2102(test) gid=2102(test) groups=2102(test),9998(ficommon)
```

Step 2 Import data again.

----End

9.10.16 Why Is the Error Message "import argparse" Displayed When the Phoenix sqlline Script Is Used?

Question

When the sqlline script is used on the client, the error message "import argparse" is displayed.

Answer

Step 1 Log in to the node where the HBase client is installed as user **root**. Perform security authentication using the **hbase** user.

Step 2 Go to the directory where the sqlline script of the HBase client is stored and run the **python3 sqlline.py** command.

----End

9.10.17 How Do I View Regions in the CLOSED State in an ENABLED Table?

Question

How do I view regions in the CLOSED state in an ENABLED table on the HBase client?

This parameter is available only in MRS 3.3.0 or later.

Procedure

Step 1 Log in to the node on which the HBase client is installed as a client installation user.

Step 2 Go to the client installation directory and configure the environment variables:

cd *Client installation directory*

source bigdata_env

Step 3 If Kerberos authentication is enabled for the cluster (the cluster is in security mode), run the following command to perform security authentication. If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), skip this step.

kinit *Component service user*

Step 4 Run the following command to check the regions in the CLOSED state in an ENABLED table:

hbase hbck -j HBase/hbase/tools/hbase-hbck2-*jar reportClosedRegions [-details] [<TABLENAME>...]

The command parameters are as follows:

- If **-details** is not specified, only the number of CLOSED regions is displayed. If **-details** is specified, the names of all CLOSED regions are displayed.

- If *TABLENAME* is not specified, all tables are queried by default.
- If "Closed region due to split" is displayed after the command is executed, the region status changes to CLOSED due to an ongoing split task. After the task is complete, the region is automatically removed from the meta table.

```
Table meta_graph is okay.
Table hbase:namespace is okay.
Table hbase:hindex is okay.
Table hbase:rsgroup is okay.
Table ns1:test1 is okay.
Table graphbaseORM_systemNotifications is okay.
Table hbase:acl is okay.
Table testComp has 1 closed regions.
Closed region due to split testComp,,1690447776336.d5f1eb4a53bf63eb688441a1e58f9835.
```

----End

9.10.18 How Can I Quickly Recover the Service When HBase Files Are Damaged Due to a Cluster Power-Off?

Symptom

The StoreFile or WAL files are damaged due to an unexpected cluster power-off. How can I quickly restore the service?

This operation is supported only for MRS 3.3.0 or later.

Cause Analysis

If the StoreFile file is damaged, related regions fail to be brought online and system keeps retry the operation. As a result, the HBase service is abnormal. If the WAL file is damaged, log splitting fails and the system keeps retry the operation. As a result, the service is abnormal. Related regions cannot be brought online and provide services for external systems.

Procedure

The HBase server provides two configuration items to determine whether to skip damaged StoreFile and WAL files. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HBase** and click **Configuration**, search for and set the parameters listed in **Table 9-39**. The parameters take effect dynamically. Save the configuration, log in to the HBase shell, and run the **update_all_config** command for the parameters to take effect.

Skipping damaged files may cause data loss. If the following parameters are set to **true** and damaged StoreFile or WAL file is skipped, **ALM-19025 Damaged StoreFile in HBase** or **ALM-19026 Damaged WAL Files in HBase** is reported, rectify the fault by referring to the alarm help.

Table 9-39 Parameters for skipping damaged files on the HBase server

Parameter	Description	Default Value
hregion.hfile.skip.errors	Whether to skip damaged HBase Files and and move them to the **/hbase/autocorrupt** or **/hbase/MasterData/autocorrupt** directory when a region is brought online. You are not advised to enable this parameter in DR scenarios.	false
hbase.hlog.split.skip.errors	Whether to skip damaged WAL files and move them to the **/hbase/corrupt** directory during log splitting.	false

9.10.19 How Do I Quickly Restore HBase After HDFS Enters the Safe Mode and the HBase Service Is Abnormal?

Symptom

For MRS 3.5.0 and later versions, the HBase service is abnormal after the HDFS enters the safe mode. How do I quickly restore the HBase service after the HDFS exits the safe mode?

Cause Analysis

After the HDFS enters the safe mode, the HBase service is abnormal and the **meta** table goes offline. After the HDFS exits the safe mode, the meta table does not go online. "No namenode available to invoke create /hbase/WALs/xxxx.meta" is displayed in the RegionServer log.

The **meta** table cannot record the online status during the online process after the HDFS fault is rectified. In addition, the number of retry times for automatic recovery of the Manager instance health check is limited. As a result, the meta table cannot be brought online. After the HDFS exits the safe mode, you need to manually restore the service.

Procedure

Step 1 Ensure that the HDFS is restored and exits the safe mode.

Step 2 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HBase**. On the **Dashboard** page, choose **More** > **Restart Service**, enter the password of the current user, and click **OK** to restart the HBase service, and then bring the **meta** table online.

----End

10 Using HDFS

10.1 Overview of HDFS File System Directories

Hadoop Distributed File System (HDFS) implements reliable and distributed read/write of massive amounts of data. HDFS is applicable to the scenario where data read/write features "write once and read multiple times". However, the write operation is performed in sequence, that is, it is a write operation performed during file creation or an adding operation performed behind the existing file. HDFS ensures that only one caller can perform write operation on a file but multiple callers can perform read operation on the file at the same time.

This section describes the directory structure in HDFS, as shown in the following table.

Table 10-1 Directory structure of the HDFS file system

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/tmp/spark2x/sparkhive-scratch	Fixed directory	Stores temporary files of metastore session in Spark2x JDBCServer.	No	Failed to run the task.
/tmp/sparkhive-scratch	Fixed directory	Stores temporary files of metastore sessions that are executed in CLI mode using Spark2x CLI.	No	Failed to run the task.
/tmp/logs/	Fixed directory	Stores container log files.	Yes	Container log files cannot be viewed.

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/tmp/carbon/	Fixed directory	Stores the abnormal data in this directory if abnormal CarbonData data exists during data import.	Yes	Error data is lost.
/tmp/Loader-${*Job name*}_${*MR job ID*}	Temporary directory	Stores the region information about Loader HBase bulkload jobs. The data is automatically deleted after the job running is completed.	No	Failed to run the Loader HBase Bulkload job.
/tmp/hadoop-omm/yarn/system/rmstore	Fixed directory	Stores the ResourceManager running information.	Yes	Status information is lost after ResourceManager is restarted.
/tmp/archived	Fixed directory	Archives the MR task logs on HDFS.	Yes	MR task logs are lost.
/tmp/hadoop-yarn/staging	Fixed directory	Stores the run logs, summary information, and configuration attributes of ApplicationMaster running jobs.	No	Services are running improperly.
/tmp/hadoop-yarn/staging/history/done_intermediate	Fixed directory	Stores temporary files in the **/tmp/hadoop-yarn/staging** directory after all tasks are executed.	No	MR task logs are lost.
/tmp/hadoop-yarn/staging/history/done	Fixed directory	The periodic scanning thread periodically moves the **done_intermediate** log file to the **done** directory.	No	MR task logs are lost.
/tmp/mr-history	Fixed directory	Stores the historical record files that are pre-loaded.	No	Historical MR task log data is lost.

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/tmp/hive-scratch	Fixed directory	Stores temporary data (such as session information) generated during Hive running.	No	Failed to run the current task.
/user/{user}/.sparkStaging	Fixed directory	Stores temporary files of the SparkJDBCServer application.	No	Failed to start the executor.
/user/spark2x/jars	Fixed directory	Stores running dependency packages of the Spark2x executor.	No	Failed to start the executor.
/user/loader	Fixed directory	Stores dirty data of Loader jobs and data of HBase jobs.	No	Failed to execute the HBase job. Or dirty data is lost.
/user/loader/etl_dirty_data_dir				
/user/loader/etl_hbase_putlist_tmp				
/user/loader/etl_hbase_tmp				
/user/oozie	Fixed directory	Stores dependent libraries required for Oozie running, which needs to be manually uploaded.	No	Failed to schedule Oozie.
/user/mapred/hadoop-mapreduce-*xxx*.tar.gz	Fixed files	Stores JAR files used by the distributed MR cache.	No	The MR distributed cache function is unavailable.
/user/hive	Fixed directory	Stores Hive-related data by default, including the depended Spark lib package and default table data storage path.	No	User data is lost.

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/user/omm-bulkload	Temporary directory	Stores HBase batch import tools temporarily.	No	Failed to import HBase tasks in batches.
/user/hbase	Temporary directory	Stores HBase batch import tools temporarily.	No	Failed to import HBase tasks in batches.
/spark2xJobHistory2x	Fixed directory	Stores Spark2x eventlog data.	No	The History Server service is unavailable, and the task fails to be executed.
/flume	Fixed directory	Stores data collected by Flume from HDFS.	No	Flume runs improperly.
/mr-history/tmp	Fixed directory	Stores logs generated by MapReduce jobs.	Yes	Log information is lost.
/mr-history/done	Fixed directory	Stores logs managed by MR JobHistory Server.	Yes	Log information is lost.
/tenant	Created when a tenant is added.	Directory of a tenant in the HDFS. By default, the system automatically creates a folder in the **/tenant** directory based on the tenant name. For example, the default HDFS storage directory for **ta1** is **tenant/ta1**. When a tenant is created for the first time, the system creates the **/tenant** directory in the HDFS root directory. You can customize the storage path.	No	The tenant account is unavailable.

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/apps{1~5}/	Fixed directory	Stores the Hive package used by WebHCat.	No	Failed to run the WebHCat tasks.
/hbase	Fixed directory	Stores HBase data.	No	HBase user data is lost.
/hbaseFileStream	Fixed directory	Stores HFS files.	No	The HFS file is lost and cannot be restored.

10.2 HDFS User Permission Management

10.2.1 Creating an HDFS Role

Scenario

This section describes how to create and configure an HDFS role on FusionInsight Manager. The HDFS role is granted the rights to read, write, and execute HDFS directories or files.

A user has the complete permission on the created HDFS directories or files, that is, the user can directly read data from and write data to as well as authorize others to access the HDFS directories or files.

NOTE

- An HDFS role can be created only in security mode.
- If the current component uses Ranger for permission control, HDFS policies must be configured based on Ranger for permission management. For details, see **Adding a Ranger Access Permission Policy for HDFS**.

Procedure

Step 1 Log in to FusionInsight Manager, and choose **System** > **Permission** > **Role**.

Step 2 On the displayed page, click **Create Role** and fill in **Role Name** and **Description**.

Step 3 Configure the resource permission. For details, see **Table 10-2**.

Table 10-2 Setting a role

Task	Operation
Setting the HDFS administrator permission	In the **Configure Resource Permission** area, choose *Name of the desired cluster* > HDFS, and select **Cluster Admin Operations**. **NOTE** The setting takes effect after the HDFS service is restarted.
Setting the permission for users to check and recover HDFS	1. In the **Configure Resource Permission** area, choose *Name of the desired cluster* > HDFS > **File System**. 2. Locate the save path of specified directories or files on HDFS. 3. In the **Permission** column of the specified directories or files, select **Read** and **Execute**.
Setting the permission for users to read directories or files of other users	1. In the **Configure Resource Permission** area, choose *Name of the desired cluster* > HDFS > **File System**. 2. Locate the save path of specified directories or files on HDFS. 3. In the **Permission** column of the specified directories or files, select **Read** and **Execute**.
Setting the permission for users to write data to files of other users	1. In the **Configure Resource Permission** area, choose *Name of the desired cluster* > HDFS > **File System**. 2. Locate the save path of specified files on HDFS. 3. In the **Permission** column of the specified files, select **Write** and **Execute**.
Setting the permission for users to create or delete sub-files or sub-directories in the directory of other users	1. In the **Configure Resource Permission** area, choose *Name of the desired cluster* > HDFS > **File System**. 2. Locate the path where the specified directory is saved in the HDFS. 3. In the **Permission** column of the specified directories, select **Write** and **Execute**.
Setting the permission for users to execute directories or files of other users	1. In the **Configure Resource Permission** area, choose *Name of the desired cluster* > HDFS > **File System**. 2. Locate the save path of specified directories or files on HDFS. 3. In the **Permission** column of the specified directories or files, select **Execute**.

Task	Operation
Setting the permission for allowing subdirectories to inherit all permissions of their parent directories	1. In the **Configure Resource Permission** area, choose *Name of the desired cluster* > HDFS > **File System**. 2. Locate the save path of specified directories or files on HDFS. 3. In the **Permission** column of the specified directories or files, select **Recursive**.

NOTE

- **File System**: HDFS directory and file permission.
- Common HDFS directories are as follows:
  - **flume**: Flume data storage directory
  - **hbase**: HBase data storage directory
  - **mr-history**: MapReduce task information storage directory
  - **tmp**: temporary data storage directory
  - **user**: user data storage directory

Step 4 Click **OK**, and return to the **Role** page.

----End

10.2.2 Configuring HDFS Directory Permission

Scenario

The permission for some HDFS directories is **777** or **750** by default, which brings potential security risks. You are advised to modify the permission for the HDFS directories after the HDFS is installed to increase user security.

Procedure

Log in to the HDFS client as the administrator and run the following command to modify the permission for the **/user** directory.

The permission is set to **1777**, that is, **1** is added to the original permission. This indicates that only the user who creates the directory can delete it.

hdfs dfs -chmod 1777 */user*

To ensure security of the system file, you are advised to harden the security for non-temporary directories. The following directories are examples:

- /user:777
- /mr-history:777
- /mr-history/tmp:777
- /mr-history/done:777

- /user/mapred:755

10.3 Using the HDFS Client

Scenario

This section describes how to use the HDFS client in an O&M scenario or service scenario.

Prerequisites

- The client has been installed.

  For example, the installation directory is **/opt/client**. The client directory in the following operations is only an example. Change it based on the actual installation directory onsite.
- Service component users have been created by the MRS cluster administrator. In security mode, machine-machine users need to download the keytab file. A human-machine user needs to change the password upon the first login. (This operation is not required in normal mode.)

Using the HDFS Client

Step 1 Install a client. For details, see **Installing a Client**.

Step 2 Log in to the node where the client is installed as the client installation user.

Step 3 Run the following command to go to the client installation directory:

cd /opt/client

Step 4 Run the following command to configure environment variables:

source bigdata_env

Step 5 If the cluster is in security mode, run the following command to authenticate the user. In normal mode, user authentication is not required.

kinit *Component service user*

Step 6 Run the HDFS Shell command. Example:

hdfs dfs -ls /

----End

Common HDFS Client Commands

The following table lists common HDFS client commands.

For more commands, see **https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/CommandsManual.html#User_Commands**.

Table 10-3 Common HDFS client commands

Command	Description	Example
hdfs dfs -mkdir *Folder name*	Used to create a folder.	**hdfs dfs -mkdir /tmp/mydir**
hdfs dfs -ls *Folder name*	Used to view a folder.	**hdfs dfs -ls /tmp**
hdfs dfs -put *Local file on the client node Specified HDFS path*	Used to upload a local file to a specified HDFS path.	**hdfs dfs -put /opt/test.txt /tmp** Upload the **/opt/test.txt** file on the client node to the **/tmp** directory of HDFS.
hdfs dfs -get *Specified file on HDFS Specified path on the client node*	Used to download the HDFS file to the specified local path.	**hdfs dfs -get /tmp/test.txt /opt/** Download the **/tmp/test.txt** file on HDFS to the **/opt** path on the client node.
hdfs dfs -rm -r -f *Specified folder on HDFS*	Used to delete a folder.	**hdfs dfs -rm -r -f /tmp/mydir**
hdfs dfs -chmod *Permission parameter File directory*	Used to configure the HDFS directory permission for a user.	**hdfs dfs -chmod 700 /tmp/test**

Client-related FAQs

1. What do I do when the HDFS client exits abnormally and error message "java.lang.OutOfMemoryError" is displayed after the HDFS client command is running?

   This problem occurs because the memory required for running the HDFS client exceeds the preset upper limit (128 MB by default). You can change the memory upper limit of the client by modifying **CLIENT_GC_OPTS** in *<Client installation path>***/HDFS/component_env**. For example, if you want to set the upper limit to 1 GB, run the following command:

   ```
   CLIENT_GC_OPTS="-Xmx1G"
   ```

   After the modification, run the following command to make the modification take effect:

   source <*Client installation path*>/**/bigdata_env**

2. How do I set the log level when the HDFS client is running?

   By default, the logs generated during the running of the HDFS client are printed to the console. The default log level is INFO. To enable the DEBUG log level for fault locating, run the following command to export an environment variable:

   export HADOOP_ROOT_LOGGER=DEBUG,console

   Then run the HDFS Shell command to generate the DEBUG logs.

If you want to print INFO logs again, run the following command:

export HADOOP_ROOT_LOGGER=INFO,console

3. How do I delete HDFS files permanently?

   HDFS provides a recycle bin mechanism. Typically, after an HDFS file is deleted, the file is moved to the recycle bin of HDFS. If the file is no longer needed and the storage space needs to be released, clear the corresponding recycle bin directory, for example, **hdfs://hacluster/user/xxx/.Trash/Current/***xxx*.

10.4 Using Hadoop from Scratch

You can use Hadoop to submit wordcount jobs. Wordcount is the most classic Hadoop job and is used to count the number of words in massive text.

Procedure

Step 1 Prepare the wordcount program.

Multiple open source Hadoop sample programs are provided, including wordcount. You can download the Hadoop sample program from **https://dist.apache.org/repos/dist/release/hadoop/common/**.

For example, choose **hadoop-***x.x.x*. On the page that is displayed, click **hadoop-***x.x.x***.tar.gz** to download it. Then, decompress it to obtain **hadoop-mapreduce-examples-***x.x.x***.jar** (the Hadoop sample program) from **hadoop-***x.x.x***\share\hadoop\mapreduce**. The **hadoop-mapreduce-examples-***x.x.x***.jar** package contains the wordcount program.

NOTE

hadoop-*x.x.x* indicates the Hadoop version. Choose a version based on your requirements.

Step 2 Prepare data files.

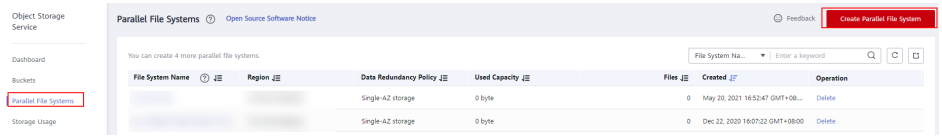
There is no format requirement for data files. Prepare one or more **.txt** files. The following are examples of the **.txt** file:

```
qwsdfhoedfrffrofhuncckgktpmhutopmma
jjpsffjfjorgjgtyiuyjmhombmbogohoyhm
jhheyeombdhuaqqiquyebchdhmamdhdemmj
doeyhjwedcrfvtgbmojiyhhqssddddddfkf
kjhhjkehdeiyrudjhfhfhffooqweopuyyyy
```

Step 3 Upload data to OBS.

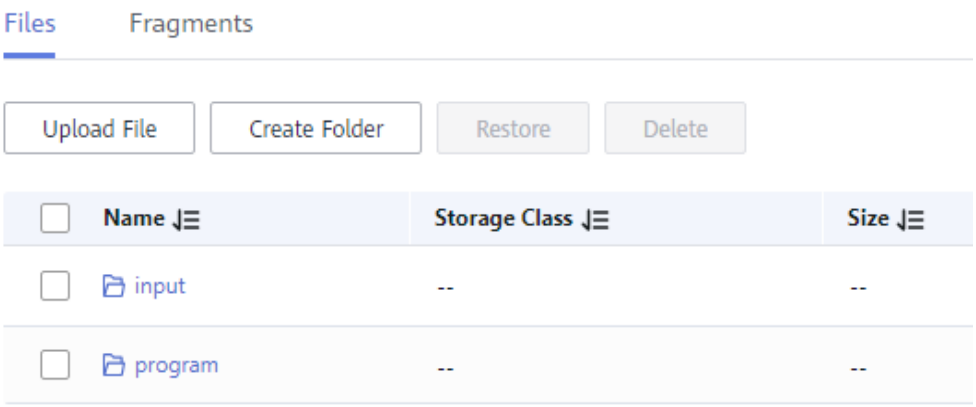
1. Log in to OBS Console.
2. Click **Parallel File System** and choose **Create Parallel File System** to create a file system named **wordcount01**.

   wordcount01 is only an example. The file system name must be globally unique. Otherwise, the parallel file system fails to be created.



3. In the OBS file system list, click **wordcount01** and choose **Files** > **Create Folder** to create the **program** and **input** folders, as shown in **Figure 10-1**.

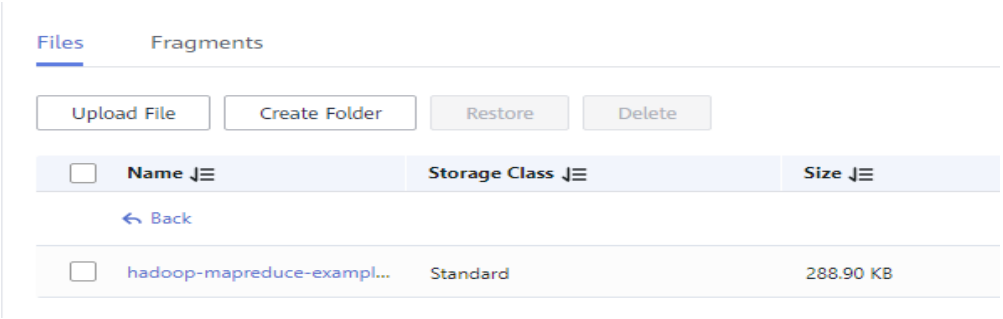
   Figure 10-1 Folder list of the wordcount01 file system

   - **program**: stores user programs.
   - **input**: stores user data files.

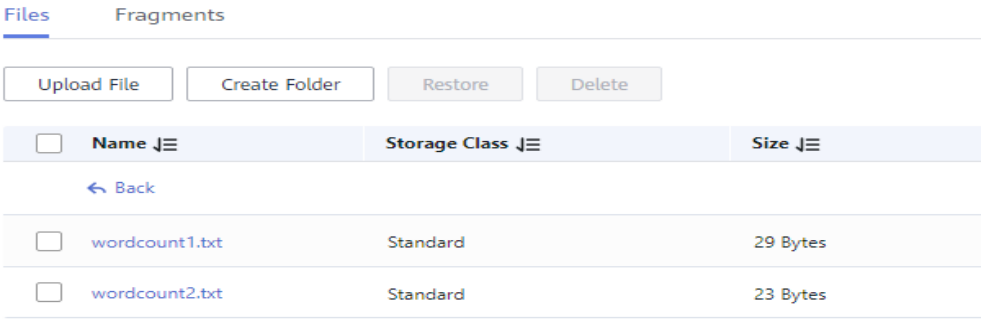
4. Go to the **program** folder, choose **Upload File** > **add file**, select the program package downloaded in **Step 1** from the local host, and click **Upload**. After the upload is complete, the page shown in **Figure 10-2** is displayed.

   Figure 10-2 Program list

5. Go to the **input** folder and upload the data file prepared in **Step 2** to the **input** folder. After the upload is complete, the page shown in **Figure 10-3** is displayed.

   Figure 10-3 Data file list

Step 4 Log in to the MRS console. In the navigation pane on the left, choose **Clusters** > **Active Clusters**. Click the cluster name. The cluster must contain Hadoop components and has been bound to an IAM agency with the OBS file system operation permission.

To view or bind an agency, perform the following steps:

1. In the **Dashboard** tab of the cluster, check whether the agency parameter has a value and whether the bound agency has the permission to operate an OBS file system.

   - If there is the agency name, the cluster has been bound to an agency.
   - If there is no agency name, go to **Step 4.2**.

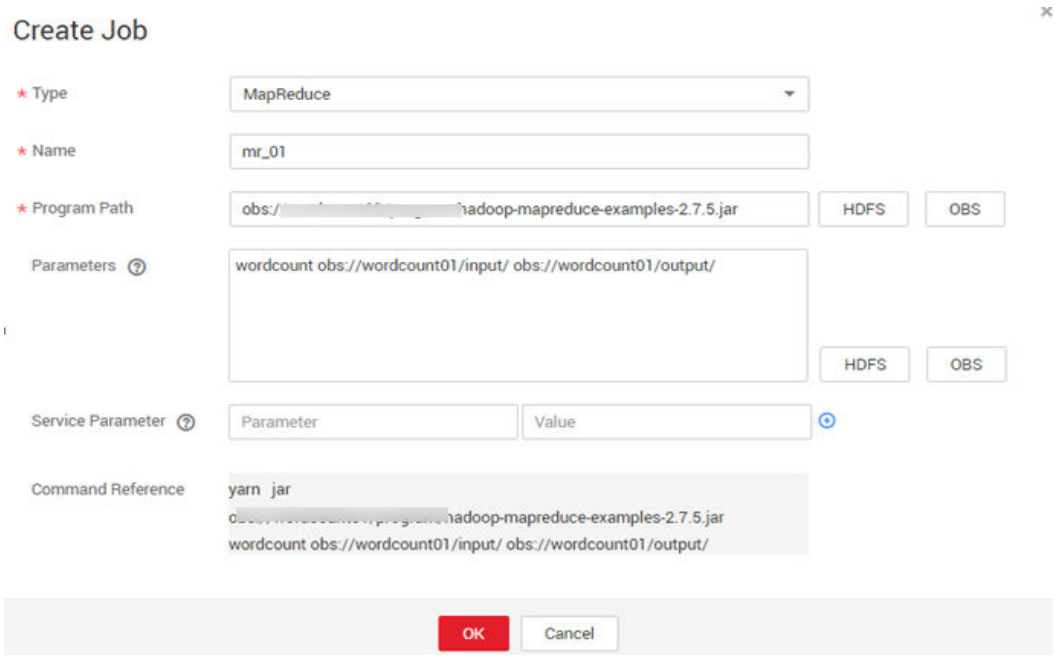
2. Click **Manage Agency** to bind an agency that has the permission to operate the OBS file system to the cluster.

   You can select the default **MRS_ECS_DEFAULT_AGENCY** agency or create an agency that has the permission to operate the OBS file system.

Step 5 Submit the wordcount job.

On the MRS console, click the **Jobs** tab and click **Create**. The **Create Job** page is displayed. For details, see **Running a MapReduce Job**.

Figure 10-4 wordcount job



- Set **Type** to **MapReduce**.
- Set **Name** to **mr_01**.
- Set the path of the executable program to the address of the program stored on the OBS, for example, **obs://wordcount01/program/hadoop-mapreduce-examples-*x.x.x*.jar**.

- Enter **wordcount obs://wordcount01/input/ obs://wordcount01/output/** in the **Parameter** pane.

  NOTE

  - Replace the OBS file system name in **obs://wordcount01/input/** with the actual name of the file system created in the environment.
  - Replace the OBS file system name in **obs://wordcount01/output/** with the name of the file system created in the actual environment. Replace **output** with a directory that does not exist based on site requirements.

- **Service Parameter** can be left blank.

A job can be submitted only when the cluster is in the **Running** state.

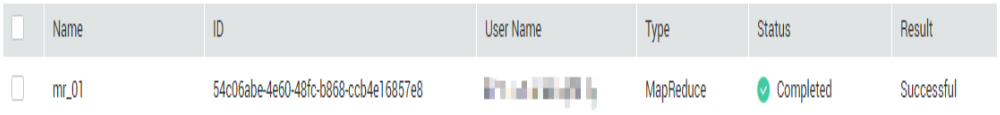
After a job is submitted successfully, it is in the **Accepted** state by default. You do not need to manually execute the job.

Step 6 View the job execution result.

1. Go to the **Jobs** tab page and check whether the job is successfully executed.

   It takes some time to run the job. After the job is complete, refresh the job list to view the job execution, as shown in **Figure 10-5**.

   Figure 10-5 Job list

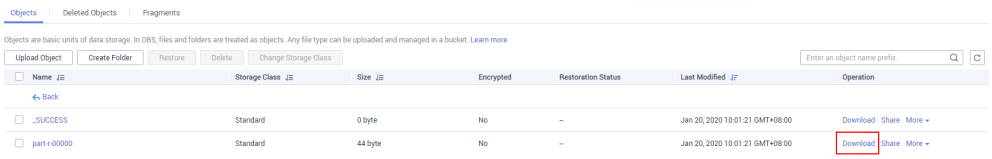
   

   Once a job has succeeded or failed, you cannot execute it again. However, you can add or copy a job, and set job parameters to submit a job again.

2. Log in to the OBS console, go to the OBS path, and view the job output information.

   You can view output files in the **output** directory created in **Step 5**. You need to download the file to the local host and open it in text format, as shown in **Figure 10-6**.

   Figure 10-6 Output file list

----End

10.5 Configuring the Recycle Bin Mechanism

Scenario

On HDFS, deleted files are moved to the recycle bin (trash can) so that the data deleted by mistake can be restored.

You can set the time threshold for storing files in the recycle bin. Once the file storage duration exceeds the threshold, it is permanently deleted from the recycle bin. If the recycle bin is cleared, all files in the recycle bin are permanently deleted.

Configuration Description

If a file is deleted from HDFS, the file is saved in the trash space rather than cleared immediately. After the aging time is due, the deleted file becomes an aging file and will be cleared based on the system mechanism or manually cleared by users.

Parameter portal:

Go to the **All Configurations** page of HDFS and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 10-4 Parameter description

Parameter	Description	Default Value
fs.trash.interval	Trash collection time, in minutes. If data in the trash station exceeds the time, the data will be deleted. Value range: 1440 to 259200	1440
fs.trash.checkpoint.interval	Interval between trash checkpoints, in minutes. The value must be less than or equal to the value of **fs.trash.interval**. The checkpoint program creates a checkpoint every time it runs and removes the checkpoint created **fs.trash.interval** minutes ago. For example, the system checks whether aging files exist every 10 minutes and deletes aging files if any. Files that are not aging are stored in the checkpoint list waiting for the next check. If this parameter is set to 0, the system does not check aging files and all aging files are saved in the system. Value range: 0 to *fs.trash.interval* **NOTE** It is not recommended to set this parameter to 0 because aging files will use up the disk space of the cluster.	60

10.6 Configuring HDFS DataNode Data Balancing

Scenario

In the HDFS cluster, unbalanced disk usage among DataNodes may occur, for example, when new DataNodes are added to the cluster. Unbalanced disk usage may result in multiple problems. For example, MapReduce applications cannot

make full use of local computing advantages, network bandwidth usage between data nodes cannot be optimal, or node disks cannot be used. Therefore, the MRS cluster administrator needs to periodically check and maintain DataNode data balance.

HDFS provides a capacity balancing program Balancer. By running Balancer, you can balance the HDFS cluster and ensure that the difference between the disk usage of each DataNode and that of the HDFS cluster does not exceed the threshold. DataNode disk usage before and after balancing is shown in **Figure 10-7** and **Figure 10-8**, respectively.

Figure 10-7 DataNode disk usage before balancing

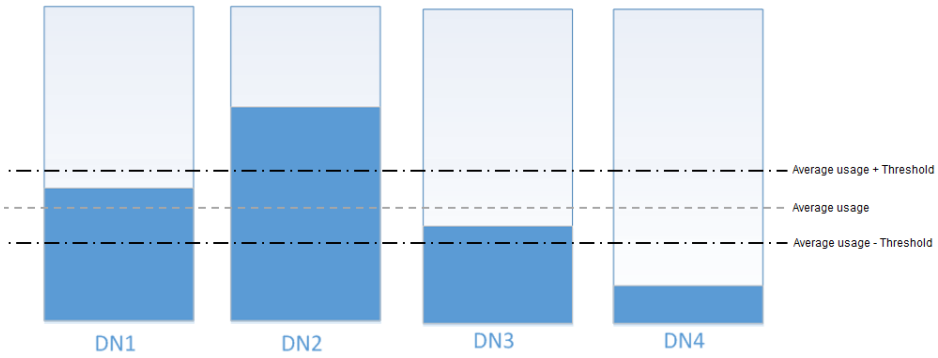
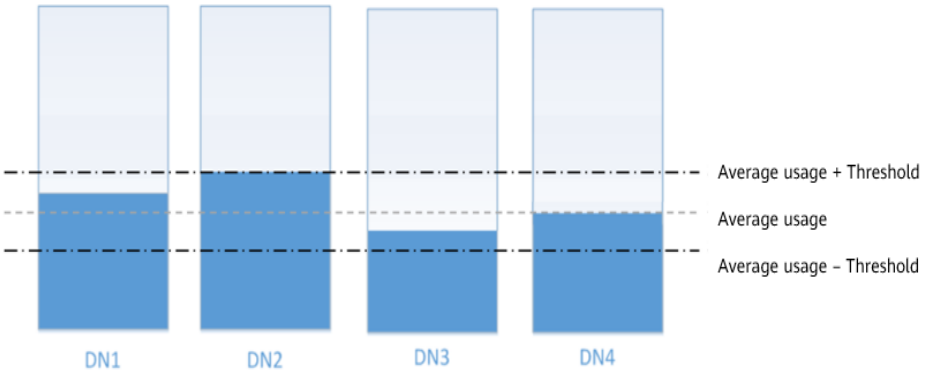


Figure 10-8 DataNode disk usage after balancing



The time of the balancing operation is affected by the following two factors:

1. Total amount of data to be migrated:

   The data volume of each DataNode must be greater than (Average usage - Threshold) x Average data volume and less than (Average usage + Threshold) x Average data volume. If the actual data volume is less than the minimum value or greater than the maximum value, imbalance occurs. The system sets the largest deviation volume on all DataNodes as the total data volume to be migrated.

2. Balancer migration is performed in sequence in iteration mode. The amount of data to be migrated in each iteration does not exceed 10 GB, and the usage of each iteration is recalculated.

Therefore, for a cluster, you can estimate the time consumed by each iteration (by observing the time consumed by each iteration recorded in balancer logs) and divide the total data volume by 10 GB to estimate the task execution time.

The balancer can be started or stopped at any time.

Impact on the System

- The balance operation occupies network bandwidth resources of DataNodes. Perform the operation during maintenance based on service requirements.
- The balance operation may affect the running services if the bandwidth traffic (the default bandwidth control is 20 MB/s) is reset or the data volume is increased.

Prerequisites

The client has been installed.

Procedure

Step 1 Log in to the node where the client is installed as a client installation user. Run the following command to switch to the client installation directory, for example, **/opt/client**:

cd /opt/client

NOTE

If the cluster is in normal mode, run the **su - omm** command to switch to user **omm**.

Step 2 Run the following command to configure environment variables:

source bigdata_env

Step 3 If the cluster is in security mode, run the following command to authenticate the HDFS identity:

kinit hdfs

Step 4 Determine whether to adjust the bandwidth control.

- If yes, go to **Step 5**.
- If no, go to **Step 6**.

Step 5 Run the following command to change the maximum bandwidth of Balancer, and then go to **Step 6**.

hdfs dfsadmin -setBalancerBandwidth *<bandwidth in bytes per second>*

<bandwidth in bytes per second> indicates the bandwidth control value, in bytes. For example, to set the bandwidth control to 20 MB/s (the corresponding value is 20971520), run the following command:

hdfs dfsadmin -setBalancerBandwidth 20971520

NOTE

- The default bandwidth control is 20 MB/s. This value is applicable to the scenario where the current cluster uses the 10GE network and services are being executed. If the service idle time window is insufficient for balance maintenance, you can increase the value of this parameter to shorten the balance time, for example, to 209715200 (200 MB/s).
- The value of this parameter depends on the networking. If the cluster load is high, you can change the value to 209715200 (200 MB/s). If the cluster is idle, you can change the value to 1073741824 (1 GB/s).
- If the bandwidth of the DataNodes cannot reach the specified maximum bandwidth, modify the HDFS parameter **dfs.datanode.balance.max.concurrent.moves** on FusionInsight Manager, and change the number of threads for balancing on each DataNode to **32** and restart the HDFS service.

Step 6 Run the following command to start the balance task:

bash /opt/client/HDFS/hadoop/sbin/start-balancer.sh -threshold *<threshold of balancer>*

-threshold specifies the deviation value of the DataNode disk usage, which is used for determining whether the HDFS data is balanced. When the difference between the disk usage of each DataNode and the average disk usage of the entire HDFS cluster is less than this threshold, the system considers that the HDFS cluster has been balanced and ends the balance task.

For example, to set deviation rate to 5%, run the following command:

bash /opt/client/HDFS/hadoop/sbin/start-balancer.sh -threshold 5

NOTE

- The preceding command executes the task in the background. You can query related logs in the **hadoop-root-balancer-***Host name***.out** log file in the **/opt/client/HDFS/hadoop/logs** directory of the host.
- To stop the balance task, run the following command:

  bash /opt/client/HDFS/hadoop/sbin/stop-balancer.sh
- If only data on some nodes needs to be balanced, you can add the **-include** parameter in the script to specify the nodes to be migrated. You can run commands to view the usage of different parameters.

  For example, run the **bash /opt/client/HDFS/hadoop/sbin/start-balancer.sh -threshold 5 -include** *IP address 1,IP address 2,IP address 3* command.
- **/opt/client** is the client installation directory. If the directory is inconsistent, replace it.
- If the command fails to be executed and the error information **Failed to APPEND_FILE /system/balancer.id** is displayed in the log, run the following command to forcibly delete **/system/balancer.id** and run the **start-balancer.sh** script again:

  hdfs dfs -rm -f /system/balancer.id

Step 7 After you run the script in **Step 6**, the **hadoop-root-balancer-***Host name***.out** log file is generated in **/opt/client/HDFS/hadoop/logs**, the client installation directory. You can view the following information in the log:

- Time Stamp
- Bytes Already Moved
- Bytes Left To Move
- Bytes Being Moved

If message "Balance took *xxx* seconds" is displayed in the log, the balancing operation is complete.

----End

Related Tasks

Enable automatic execution of the balance task

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > *Name of the desired cluster* > **Services** > **HDFS** > **Configurations**, select **All Configurations**, search for the following parameters, and change the parameter values.

- **dfs.balancer.auto.enable** indicates whether to enable automatic balance task execution. The default value **false** indicates that automatic balance task execution is disabled. The value **true** indicates that automatic execution is enabled.
- **dfs.balancer.auto.cron.expression** indicates the task execution time. The default value **0 1 * * 6** indicates that the task is executed at 01:00 every Saturday. This parameter is valid only when the automatic execution is enabled.

  Table 10-5 describes the expression for modifying this parameter. ***** indicates consecutive time segments.

  Table 10-5 Parameters in the execution expression

Column	Description
1	Minute. The value ranges from 0 to 59.
2	Hour. The value ranges from 0 to 23.
3	Date. The value ranges from 1 to 31.
4	Month. The value ranges from 1 to 12.
5	Week. The value ranges from 0 to 6. **0** indicates Sunday.

- **dfs.balancer.auto.stop.cron.expression** indicates the task ending time. The default value is empty, indicating that the running balance task is not automatically stopped. For example, **0 5 * * 6** indicates that the balance task is stopped at 05:00 every Saturday. This parameter is valid only when the automatic execution is enabled.

  Table 10-5 describes the expression for modifying this parameter. ***** indicates consecutive time segments.

Step 3 Running parameters of the balance task that is automatically executed are shown in Table 10-6.

Table 10-6 Running parameters of the automatic balancer

Parameter	Parameter description	Default Value
dfs.balancer.auto.threshold	Specifies the balancing threshold of the disk capacity percentage. This parameter is valid only when **dfs.balancer.auto.enable** is set to **true**.	10
dfs.balancer.auto.exclude.datanodes	Specifies the list of DataNodes on which automatic disk balancing is not required. This parameter is valid only when **dfs.balancer.auto.enable** is set to **true**.	The value is left blank by default.
dfs.balancer.auto.bandwidthPerSec	Specifies the maximum bandwidth (MB/s) of each DataNode for load balancing.	20
dfs.balancer.auto.maxIdleIterations	Specifies the maximum number of consecutive idle iterations of Balancer. An idle iteration is an iteration without moving blocks. When the number of consecutive idle iterations reaches the maximum number, the balance task ends. The value **-1** indicates infinity.	5
dfs.balancer.auto.maxDataNodesNum	Controls the number of DataNodes that perform automatic balance tasks. Assume that the value of this parameter is *N*. If *N* is greater than 0, data is balanced between *N* DataNodes with the highest percentage of remaining space and *N* DataNodes with the lowest percentage of remaining space. If *N* is 0, data is balanced among all DataNodes in the cluster.	5

Step 4 Click **Save** to make configurations take effect. You do not need to restart the HDFS service.

Go to the **/var/log/Bigdata/hdfs/nn/hadoop-omm-balancer-***Host name***.log** file to view the task execution logs saved in the active NameNode.

----End

10.7 Configuring HDFS DiskBalancer

Scenario

DiskBalancer is an online disk balancer that balances disk data on running DataNodes based on various indicators. It works in the similar way of the HDFS Balancer. The difference is that HDFS Balancer balances data between DataNodes, while HDFS DiskBalancer balances data among disks on a single DataNode.

Data among disks may be unevenly distributed if a large number of files have been deleted from a cluster running for a long time, or disk capacity expansion is performed on a node in the cluster. Uneven data distribution may deteriorate the concurrent read/write performance of the HDFS, or cause service failure due to inappropriate HDFS write policies. In this case, the data density among disks on a node needs to be balanced to prevent heterogeneous small disks from becoming the performance bottleneck of the node.

Configuration Description

Go to the **All Configurations** page of HDFS and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 10-7 Parameter description

Parameter	Description	Default Value
dfs.disk.balancer.auto.enabled	Indicates whether to enable the HDFS DiskBalancer function. The default value is **false**, indicating that this function is disabled.	false
dfs.disk.balancer.auto.cron.expression	CRON expression of the HDFS disk balancing operation, which is used to control the start time of the balancing operation. This parameter is valid only when **dfs.disk.balancer.auto.enabled** is set to **true**. The default value is **0 1 * * 6**, indicating that tasks are executed at 01:00 every Saturday. For details about cron expression, see **Table 10-8**. The default value indicates that the DiskBalancer check is executed at 01:00 every Saturday.	0 1 * * 6
dfs.disk.balancer.max.disk.throughputInMBperSec	Specifies the maximum disk bandwidth that can be used for disk data balancing. The unit is MB/s, and the default value is **10**. Set this parameter based on the actual disk conditions of the cluster.	10
dfs.disk.balancer.max.disk.errors	Specifies the maximum number of errors that are allowed in a specified movement process. If the value exceeds this threshold, the movement fails.	5

Parameter	Description	Default Value
dfs.disk.balancer.block.tolerance.percent	Specifies the difference threshold between the data storage capacity and perfect status of each disk during data balancing among disks. For example, the ideal data storage capacity of each disk is 1 TB, and this parameter is set to **10**. When the data storage capacity of the target disk reaches 900 GB, the storage status of the disk is considered as perfect. Value range: 1 to 100.	10
dfs.disk.balancer.plan.threshold.percent	Specifies the data density difference that is allowed between two disks during disk data balancing. If the absolute value of the data density difference between any two disks exceeds the threshold, data balancing is required. Value range: 1 to 100.	10
dfs.disk.balancer.top.nodes.number	Specifies the top *N* nodes whose disk data needs to be balanced in the cluster.	5

To use this function, set **dfs.disk.balancer.auto.enabled** to **true** and configure a proper CRON expression. Set other parameters based on the cluster status.

Table 10-8 CRON expressions

Column	Description
1	Minute. The value ranges from 0 to 59.
2	Hour. The value ranges from 0 to 23.
3	Date. The value ranges from 1 to 31.
4	Month. The value ranges from 1 to 12.
5	Week. The value ranges from 0 to 6. **0** indicates Sunday.

Use Restrictions

1. Data can only be moved between disks of the same type. For example, data can only be moved between SSDs or between DISKs.
2. Enabling this function occupies disk I/O resources and network bandwidth resources of involved nodes. Enable this function in off-peak hours.
3. The DataNodes specified by the **dfs.disk.balancer.top.nodes.number** parameter are frequently calculated. Therefore, set the parameter to a small value.

4. Commands for using the DiskBalancer function on the HDFS client are as follows:

Table 10-9 DiskBalancer commands

Syntax	Description
hdfs diskbalancer -report -top <N>	Set *N* to an integer greater than 0. This command can be used to query the top *N* nodes that require disk data balancing in the cluster.
hdfs diskbalancer -plan <Hostname|IP Address>	This command can be used to generate a JSON file based on the DataNode. The file contains information about the source disk, target disk, and blocks to be moved. In addition, this command can be used to specify other parameters such as the network bandwidth.
hdfs diskbalancer -query <Hostname:$dfs.datanode.ipc.port>	The default port number of the cluster is 9867. This command is used to query the running status of the DiskBalancer task on the current node.
hdfs diskbalancer -execute <planfile>	In this command, **planfile** indicates the JSON file generated in the second command. Use the absolute path.
hdfs diskbalancer -cancel <planfile>	This command is used to cancel the running planfile. Use the absolute path.

NOTE

- Users running this command on the client must have the **supergroup** permission. You can use the system user **hdfs** of the HDFS service. Alternatively, you can create a user with the **supergroup** permission in the cluster and then run the command.
- Only formats and usage of commands are provided in **Table 10-9**. For more parameters to be configured for each command, run the **hdfs diskbalancer -help <command>** command to view detailed information.
- When you troubleshoot performance problems during the cluster O&M, check whether the HDFS disk balancing occurs in the event information of the cluster. If yes, check whether DiskBalancer is enabled in the cluster.
- After the automatic DiskBalancer function is enabled, the ongoing task stops only after the current data balancing is complete. The task cannot be canceled during the balancing.
- You can manually specify certain nodes for data balancing on the client.

10.8 Configuring HDFS Mover

Scenario

Mover is a new data migration tool whose working mode is similar to that of the HDFS Balancer. Mover can redistribute data in the cluster based on the configured data storage policy.

Use Mover to periodically check whether the specified HDFS file or directory in the HDFS file system meets the preset storage policy. If not, migrate data to make them meet the policy.

Configuration Description

Go to the **All Configurations** page of HDFS and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 10-10 Parameter description

Parameter	Description	Default Value
dfs.mover.auto.enable	Specifies whether to enable the data replica migration function. This function supports multiple modes. The default value is **false**, indicating that this function is disabled.	false
dfs.mover.auto.cron.expression	Specifies the CRON expression for HDFS automatic data migration, and is used to control the start time of data migration. This parameter is valid only when **dfs.mover.auto.enable** is set to **true**. The default value is **0 * * * ***, indicating that the task is executed on the hour. For details about CRON expression, see **Table 10-11**.	0 * * * *
dfs.mover.auto.hdfsfiles_or_dirs	Specifies HDFS file and directory lists that implement automatic replica migration in specified clusters. Multiple values are separated by space. This parameter is valid only when **dfs.mover.auto.enable** is set to **true**.	-

Table 10-11 CRON expressions

Column	Description
1	Minute. The value ranges from 0 to 59.
2	Hour. The value ranges from 0 to 23.
3	Date. The value ranges from 1 to 31.

Column	Description
4	Month. The value ranges from 1 to 12.
5	Week. The value ranges from 0 to 6. **0** indicates Sunday.

Use Restrictions

Run the command on the HDFS client to enable the mover function. The command format is as follows:

hdfs mover -p *<Full path or directory path of an HDFS file >*

NOTE

Users running this command on the client must have the **supergroup** permission. You can use the system user **hdfs** of the HDFS service. Alternatively, you can create a user with the **supergroup** permission in the cluster and then run the command.

10.9 Configuring HDFS NodeLabel

Scenario

You need to configure the nodes for storing HDFS file data blocks based on data features. You can configure a label expression to an HDFS directory or file and assign one or more labels to a DataNode so that file data blocks can be stored on specified DataNodes.

If the label-based data block placement policy is used for selecting DataNodes to store the specified files, the DataNode range is specified based on the label expression. Then proper nodes are selected from the specified range.

- Scenario 1: DataNodes partitioning scenario

  Scenario description:

  When different application data is required to run on different nodes for separate management, label expressions can be used to achieve separation of different services, storing specified services on corresponding nodes.

  By configuring the NodeLabel feature, you can perform the following operations:

  - Store data in **/HBase** to DN1, DN2, DN3, and DN4.
  - Store data in **/Spark** to DN5, DN6, DN7, and DN8.

Figure 10-9 DataNode partitioning scenario

NOTE

- Run the **hdfs nodelabel -setLabelExpression -expression 'LabelA[fallback=NONE]' -path /Hbase** command to set an expression for the **Hbase** directory. As shown in **Figure 10-9**, the data block replicas of files in the **/Hbase** directory are placed on the nodes labeled with the **LabelA**, that is, DN1, DN2, DN3, and DN4. Similarly, run the **hdfs nodelabel -setLabelExpression -expression 'LabelB[fallback=NONE]' -path /Spark** command to set an expression for the Spark directory. Data block replicas of files in the **/Spark** directory can be placed only on nodes labeled with **LabelB**, that is, DN5, DN6, DN7, and DN8.
- For details about how to set labels for a data node, see **Configuration Description**.
- If multiple racks are available in one cluster, it is recommended that DataNodes of these racks should be available under each label, to ensure reliability of data block placement.

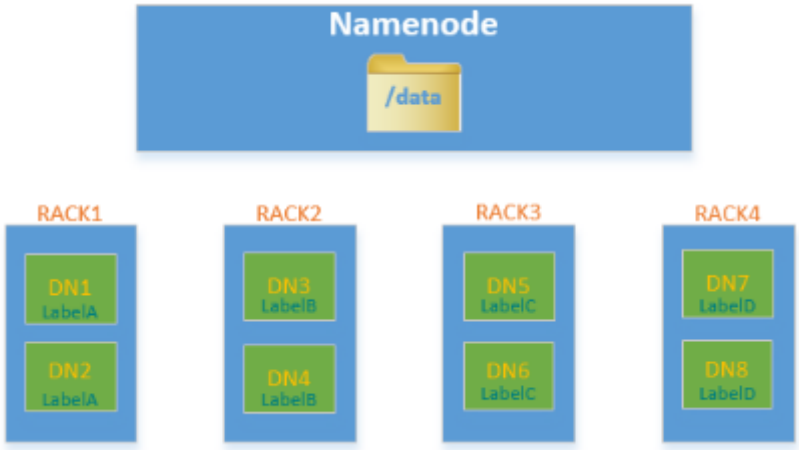
- Scenario 2: Specifying replica location when there are multiple racks

  Scenario description:

  In a heterogeneous cluster, customers need to allocate certain nodes with high availability to store important commercial data. Label expressions can be used to specify replica location so that the replica can be placed on a high reliable node.

  Data blocks in the **/data** directory have three replicas by default. In this case, at least one replica is stored on a node of RACK1 or RACK2 (nodes of RACK1 and RACK2 are high reliable), and the other two are stored separately on the nodes of RACK3 and RACK4.

Figure 10-10 Scenario example



NOTE

Run the **hdfs nodelabel -setLabelExpression -expression 'LabelA|| LabelB[fallback=NONE],LabelC,LabelD' -path /data** command to set an expression for the **/data** directory.

When data is to be written to the **/data** directory, at least one data block replica is stored on a node labeled with the LabelA or LabelB, and the other two data block replicas are stored separately on the nodes labeled with the LabelC and LabelD.

Configuration Description

- DataNode label configuration

  Go to the **All Configurations** page of HDFS and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

  Table 10-12 Parameter description

Parameter	Description	Default Value
dfs.block.replicator.classname	Used to configure the DataNode policy of HDFS. To enable the NodeLabel function, set this parameter to **org.apache.hadoop.hdfs.server.blockmanagement.BlockPlacementPolicyWithNodeLabel**.	org.apache.hadoop.hdfs.server.blockmanagement.AvailableSpaceBlockPlacementPolicy

Parameter	Description	Default Value
host2tags	Used to configure a mapping between a DataNode host and a label. The host name can be configured with an IP address extension expression (for example, **192.168.1.[1-128]** or **192.168.[2-3].[1-128]**) or a regular expression (for example, **/datanode-[123]/** or **/datanode-\d{2}/**) starting and ending with a slash (/). The label configuration name cannot contain the following characters: = / \ **Note**: The IP address must be a service IP address.	-

NOTE

- The **host2tags** configuration item is described as follows:

  Assume there are 20 DataNodes which range from dn-1 to dn-20 in a cluster and the IP addresses of clusters range from 10.1.120.1 to 10.1.120.20. The value of **host2tags** can be represented in either of the following methods:

  Regular expression of the host name

  /dn-\d/ = label-1 indicates that the labels corresponding to dn-1 to dn-9 are label-1, that is, dn-1 = label-1, dn-2 = label-1, ..., dn-9 = label-1.

  /dn-((1[0-9]$)|(20$))/ = label-2 indicates that the labels corresponding to dn-10 to dn-20 are label-2, that is, dn-10 = label-2, dn-11 = label-2, ...dn-20 = label-2.

  IP address range expression

  10.1.120.[1-9] = label-1 indicates that the labels corresponding to 10.1.120.1 to 10.1.120.9 are label-1, that is, 10.1.120.1 = label-1, 10.1.120.2 = label-1, ..., and 10.1.120.9 = label-1.

  10.1.120.[10-20] = label-2 indicates that the labels corresponding to 10.1.120.10 to 10.1.120.20 are label-2, that is, 10.1.120.10 = label-2, 10.1.120.11 = label-2, ..., and 10.1.120.20 = label-2.

- Label-based data block placement policies are applicable to capacity expansion and reduction scenarios.

  A newly added DataNode will be assigned a label if the IP address of the DataNode is within the IP address range in the **host2tags** configuration item or the host name of the DataNode matches the host name regular expression in the **host2tags** configuration item.

  For example, the value of **host2tags** is **10.1.120.[1-9] = label-1**, but the current cluster has only three DataNodes: 10.1.120.1 to 10.1.120.3. If DataNode 10.1.120.4 is added for capacity expansion, the DataNode is labeled as label-1. If the 10.1.120.3 DataNode is deleted or out of the service, no data block will be allocated to the node.

- Set label expressions for directories or files.
  - On the HDFS parameter configuration page, configure **path2expression** to configure the mapping between HDFS directories and labels. If the configured HDFS directory does not exist, the configuration can succeed. When a directory with the same name as the HDFS directory is created manually, the configured label mapping relationship will be inherited by the directory within 30 minutes. After a labeled directory is deleted, a

new directory with the same name as the deleted one will inherit its mapping within 30 minutes.

- For details about configuring items using commands, see the **hdfs nodelabel -setLabelExpression** command.
- To set label expressions using the Java API, invoke the **setLabelExpression(String src, String labelExpression)** method using the instantiated object NodeLabelFileSystem. *src* indicates a directory or file path on HDFS, and **labelExpression** indicates the label expression.

- After the NodeLabel is enabled, you can run the **hdfs nodelabel -listNodeLabels** command to view the label information of each DataNode.

Block Replica Location Selection

Nodelabel supports different placement policies for replicas. The expression **label-1,label-2,label-3** indicates that three replicas are respectively placed in DataNodes containing label-1, label-2, and label-3. Different replica policies are separated by commas (,).

If you want to place two replicas in DataNode with label-1, set the expression as follows: **label-1[replica=2],label-2,label-3**. In this case, if the default number of replicas is 3, two nodes with label-1 and one node with label-2 are selected. If the default number of replicas is 4, two nodes with label-1, one node with label-2, and one node with label-3 are selected. Note that the number of replicas is the same as that of each replica policy from left to right. However, the number of replicas sometimes exceeds the expressions. If the default number of replicas is 5, the extra replica is placed on the last node, that is, the node labeled with label-3.

When the ACLs function is enabled and the user does not have the permission to access the labels used in the expression, the DataNode with the label is not selected for the replica.

Deletion of Redundant Block Replicas

If the number of block replicas exceeds the value of **dfs.replication** (number of file replicas specified by the user), HDFS will delete redundant block replicas to ensure cluster resource usage.

The deletion rules are as follows:

- Preferentially delete replicas that do not meet any expression.

  For example: The default number of file replicas is **3**.

  The label expression of **/test** is **LA[replica=1],LB[replica=1],LC[replica=1]**.

  The file replicas of **/test** are distributed on four nodes (D1 to D4), corresponding to labels (LA to LD).

  ```
  D1:LA
  D2:LB
  D3:LC
  D4:LD
  ```

  Then, block replicas on node D4 will be deleted.

- If all replicas meet the expressions, delete the redundant replicas which are beyond the number specified by the expression.

  For example: The default number of file replicas is **3**.

The label expression of **/test** is **LA[replica=1],LB[replica=1],LC[replica=1]**.

The file replicas of **/test** are distributed on the following four nodes, corresponding to the following labels.

```
D1:LA
D2:LA
D3:LB
D4:LC
```

Then, block replicas on node D1 or D2 will be deleted.

- If a file owner or group of a file owner cannot access a label, preferentially delete the replica from the DataNode mapped to the label.

Example of label-based block placement policy

Assume that there are six DataNodes, namely, dn-1, dn-2, dn-3, dn-4, dn-5, and dn-6 in a cluster and the corresponding IP address range is 10.1.120.[1-6]. Six directories must be configured with label expressions. The default number of block replicas is **3**.

- The following provides three expressions of the DataNode label in **host2labels** file. The three expressions have the same function.
  - Regular expression of the host name

    ```
    /dn-[1456]/ = label-1,label-2
    /dn-[26]/ = label-1,label-3
    /dn-[3456]/ = label-1,label-4
    /dn-5/ = label-5
    ```

  - IP address range expression

    ```
    10.1.120.[1-6] = label-1
    10.1.120.1 = label-2
    10.1.120.2 = label-3
    10.1.120.[3-6] = label-4
    10.1.120.[4-6] = label-2
    10.1.120.5 = label-5
    10.1.120.6 = label-3
    ```

  - Common host name expression

    ```
    /dn-1/ = label-1, label-2
    /dn-2/ = label-1, label-3
    /dn-3/ = label-1, label-4
    /dn-4/ = label-1, label-2, label-4
    /dn-5/ = label-1, label-2, label-4, label-5
    /dn-6/ = label-1, label-2, label-3, label-4
    ```

- The label expressions of the directories are set as follows:

  ```
  /dir1 = label-1
  /dir2 = label-1 && label-3
  /dir3 = label-2 || label-4[replica=2]
  /dir4 = (label-2 || label-3) && label-4
  /dir5 = !label-1
  /sdir2.txt = label-1 && label-3[replica=3,fallback=NONE]
  /dir6 = label-4[replica=2],label-2
  ```

  NOTE

  For details about the label expression configuration, see the **hdfs nodelabel -setLabelExpression** command.

  The file data block storage locations are as follows:

  - Data blocks of files in the **/dir1** directory can be stored on any of the following nodes: dn-1, dn-2, dn-3, dn-4, dn-5, and dn-6.
  - Data blocks of files in the **/dir2** directory can be stored on the dn-2 and dn-6 nodes. The default number of block replicas is **3**. The expression

matches only two DataNodes. The third replica will be stored on one of the remaining nodes in the cluster.

- Data blocks of files in the **/dir3** directory can be stored on any three of the following nodes: dn-1, dn-3, dn-4, dn-5, and dn-6.
- Data blocks of files in the **/dir4** directory can be stored on the dn-4, dn-5, and dn-6 nodes.
- Data blocks of files in the **/dir5** directory do not match any DataNode and will be stored on any three nodes in the cluster, which is the same as the default block selection policy.
- For the data blocks of the **/sdir2.txt** file, two replicas are stored on the dn-2 and dn-6 nodes. The left one is not stored in the node because **fallback=NONE** is enabled.
- Data blocks of the files in the **/dir6** directory are stored on the two nodes with label-4 selected from dn-3, dn-4, dn-5, and dn-6 and another node with label-2. If the specified number of file replicas in the **/dir6** directory is more than 3, the extra replicas will be stored on a node with label-2.

Restrictions

In configuration files, **key** and **value** are separated by equation signs (=), colons (:), and whitespace. Therefore, the host name of the **key** cannot contain these characters because these characters may be considered as separators.

10.10 Configuring Memory Management

Scenario

In HDFS, each file object needs to register corresponding information in the NameNode and occupies certain storage space. As the number of files increases, if the original memory space cannot store the corresponding information, you need to change the memory size.

Configuration Description

Navigation path for setting parameters:

Go to the **All Configurations** page of HDFS by referring to **Modifying Cluster Service Configuration Parameters**.

Table 10-13 Parameter description

Parameter	Description	Default Value
GC_PROFILE	The NameNode memory size depends on the size of FsImage, which can be calculated based on the following formula: FsImage size = Number of files x 900 bytes. You can estimate the memory size of the NameNode of HDFS based on the calculation result. The value range of this parameter is as follows: • **high**: 4 GB • **medium**: 2 GB • **low**: 256 MB • **custom**: The memory size can be set according to the data size in GC_OPTS.	custom
GC_OPTS	JVM parameter used for garbage collection (GC). This parameter is valid only when **GC_PROFILE** is set to **custom**. Ensure that the **GC_OPT** parameter is set correctly. Otherwise, the process will fail to be started. **NOTICE** Exercise caution when you modify the configuration. If the configuration is incorrect, the services are unavailable.	-Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=128M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -Djdk.tls.ephemeralDHKeySize=2048

10.11 Configuring ulimit for HBase and HDFS

Symptom

When an HDFS file is opened and the number of handles is limited, the following error occurs:

```
IOException (Too many open files)
```

Procedure

You can contact the cluster administrator to increase the number of handles for each user. This is a configuration on the OS instead of that for HBase or HDFS. It is recommended that the cluster administrator set the number of handles based on the service volume of HBase and HDFS and the permissions of each user. If a user needs to frequently perform many operations on the HDFS with heavy service volume, set a large number of handles for the user to avoid the preceding error.

Step 1 Log in to operating systems of all nodes or clients in the cluster as user **root** and go to the **/etc/security** directory.

Step 2 Run the following command to edit the **limits.conf** file:

vi limits.conf

Added the following contents:

```
hdfs -       nofile  32768
hbase -      nofile  32768
```

hdfs and **hbase** indicate the OS user names used in services.

NOTE

- Only user **root** has the permission to edit the **limits.conf** file.
- If the modification does not take effect, check whether there are other **nofile** values for the OS user in the **/etc/security/limits.d** directory. Such values may overwrite the value configured in **/etc/security/limits.conf**.
- If a user needs to perform operations on HBase, it is recommended that the handle count of the user be set to more than 10,000. If a user needs to perform operations on HDFS, it is recommended that the handle count be set based on the service volume. (A small value is not recommended.) If a user needs to perform operations on HBase and HDFS, it is recommended that the handle count be set to a large value, for example, 32,768.

Step 3 You can run the following command to view the maximum number of handles of a user:

su - *user_name*

ulimit -n

The command output displays the maximum number of handles of the user. The following is an example.

```
8194
```

----End

10.12 Configuring the Number of Files in a Single HDFS Directory

Scenario

Generally, multiple services are deployed in a cluster, and the storage of most services depends on the HDFS file system. Different components such as Spark and Yarn or clients are constantly writing files to the same HDFS directory when the cluster is running. However, the number of files in a single directory in HDFS is limited. Users must plan to prevent excessive files in a single directory and task failure.

You can set the number of files in a single directory using the **dfs.namenode.fs-limits.max-directory-items** parameter in HDFS.

Procedure

Step 1 Go to the **All Configurations** page of HDFS by referring to **Modifying Cluster Service Configuration Parameters**.

Step 2 Search for the configuration item **dfs.namenode.fs-limits.max-directory-items**.

Table 10-14 Parameter description

Parameter	Description	Default Value
dfs.namenode.fs-limits.max-directory-items	Maximum number of items in a directory Value range: 1 to 6,400,000	1048576

Step 3 Set the maximum number of files that can be stored in a single HDFS directory. Save the modified configuration. Restart the expired service or instance for the configuration to take effect.

NOTE

Plan data storage in advance based on time and service type categories to prevent excessive files in a single directory. You are advised to use the default value, which is about 1 million pieces of data in a single directory.

----End

10.13 Enterprise-Class Enhancements of HDFS

10.13.1 Configuring the HDFS Quick File Close Function

Scenario

By default, an HDFS file can be closed only if all blocks are reported (in the **COMPLETED** state). Therefore, the write performance of HDFS is affected by waiting for DataNode blocks and NameNode processing blocks to be reported. For a cluster with heavy load, the waiting consumption has a great impact on the cluster. You can configure the **dfs.namenode.file.close.num-committed-allowed** parameter of HDFS to close files in advance to improve data write performance. However, data may fail to be read because the block cannot be found or the data block information recorded in the NameNode metadata is inconsistent with that stored in the DataNode. Therefore, this feature does not apply to the scenario where data is read immediately after being written. Exercise caution when using this feature.

NOTE

This section applies to MRS 3.2.0-LTS.1 or later.

Procedure

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **HDFS** and click the **Configurations** tab and then **All Configurations**.

Step 3 Search for and modify the **dfs.namenode.file.close.num-committed-allowed** parameter. For more information, see the following table.

Parameter	Description
dfs.namenode.file.close.num-committed-allowed	Maximum number of blocks in the **COMMITTED** state in the file to be closed. The default value is 0, indicating that this feature is disabled. If this feature is enabled, the recommended value is **1** or **2**. For example, if this parameter is set to **1**, it indicates that a file can be closed without waiting for status of the last block status to change to **COMPLETED**.

Step 4 Save the configuration.

Step 5 On the **Instance** page of HDFS, select the active and standby NameNode instances, choose **More** > **Instance Rolling Restart**, and wait until the rolling restart is complete.

----End

10.13.2 Configuring Replica Replacement Policy for Heterogeneous Capacity Among DataNodes

Scenario

By default, NameNode randomly selects a DataNode to write files. If the disk capacity of some DataNodes in a cluster is inconsistent (the total disk capacity of some nodes is large and of some nodes is small), the nodes with small disk capacity will be fully written. To resolve this problem, change the default disk selection policy for data written to DataNode to the available space block policy. This policy increases the probability of writing data blocks to the node with large available disk space. This ensures that the node usage is balanced when disk capacity of DataNodes is inconsistent.

Impact on the System

The disk selection policy is changed to **org.apache.hadoop.hdfs.server.blockmanagement.AvailableSpaceBlockPlacementPolicy**. It is proven that the HDFS file write performance optimizes by 3% after the modification.

NOTE

The default replica storage policy of the NameNode is as follows:

1. First replica: stored on the node where the client resides.
2. Second replica: stored on DataNodes of the remote rack.
3. Third replica: stored on different nodes of the same rack for the node where the client resides.

If there are more replicas, randomly store them on other DataNodes.

The replica selection mechanism (**org.apache.hadoop.hdfs.server.blockmanagement.AvailableSpaceBlockPlacementPolicy**) is as follows:

1. First replica: stored on the DataNode where the client resides (the same as the default storage policy).
2. Second replica:
   - When selecting a storage node, select two data nodes that meet the requirements.
   - Compare the disk usages of the two DataNodes. If the difference is smaller than 5%, store the replicas to the first node.
   - If the difference exceeds 5%, there is a 60% probability (specified by **dfs.namenode.available-space-block-placement-policy.balanced-space-preference-fraction** and default value is **0.6**) that the replica is written to the node whose disk space usage is low.
3. As for the storage of the third replica and subsequent replicas, refer to that of the second replica.

Prerequisites

The total disk capacity deviation of DataNodes in the cluster cannot exceed 100%.

Procedure

Step 1 Go to the **All Configurations** page of HDFS by referring to **Modifying Cluster Service Configuration Parameters**.

Step 2 Modify the disk selection policy parameters when HDFS writes data. Search for the **dfs.block.replicator.classname** parameter and change its value to **org.apache.hadoop.hdfs.server.blockmanagement.AvailableSpaceBlockPlacementPolicy**.

Step 3 Save the modified configuration. Restart the expired service or instance for the configuration to take effect.

----End

10.13.3 Configuring Reserved Percentage of Disk Usage on DataNodes

Scenario

When the Yarn local directory and DataNode directory are on the same disk, the disk with larger capacity can run more tasks. Therefore, more intermediate data is stored in the Yarn local directory.

Currently, you can set **dfs.datanode.du.reserved** to configure the absolute value of the reserved disk space on DataNodes. A small value cannot meet the requirements of a disk with large capacity. However, configuring a large value for a disk with same capacity wastes a lot of disk space.

To avoid this problem, a new parameter **dfs.datanode.du.reserved.percentage** is introduced to configure the reserved percentage of the disk space.

NOTE

- If **dfs.datanode.du.reserved.percentage** and **dfs.datanode.du.reserved** are configured at the same time, the larger value of the reserved disk space calculated using the two parameters is used as the reserved space of the data nodes.
- You are advised to set **dfs.datanode.du.reserved** or **dfs.datanode.du.reserved.percentage** based on the actual disk space.

Configuration Description

Go to the **All Configurations** page of HDFS and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 10-15 Parameter description

Parameter	Description	Default Value
dfs.datanode.du.reserved.percentage	Indicates the percentage of the reserved disk space on DataNodes. The DataNode permanently reserves the disk space calculated using this percentage. The value is an integer ranging from 0 to 100.	10

10.13.4 Configuring the NameNode Blacklist

Scenario

In the existing default DFSclient failover proxy provider, if a NameNode in a process is faulty, all HDFS client instances in the same process attempt to connect to the NameNode again. As a result, the application waits for a long time and timeout occurs.

When clients in the same JVM process connect to the NameNode that cannot be accessed, the system is overloaded. The NameNode blacklist is equipped with the MRS cluster to avoid this problem.

In the new Blacklisting DFSClient failover provider, the faulty NameNode is recorded in a list. The DFSClient then uses the information to prevent the client from connecting to such NameNodes again. This function is called NameNode blacklisting.

For example, there is a cluster with the following configurations:

namenode: nn1, nn2

dfs.client.failover.connection.retries: 20

Processes in a single JVM: 10 clients

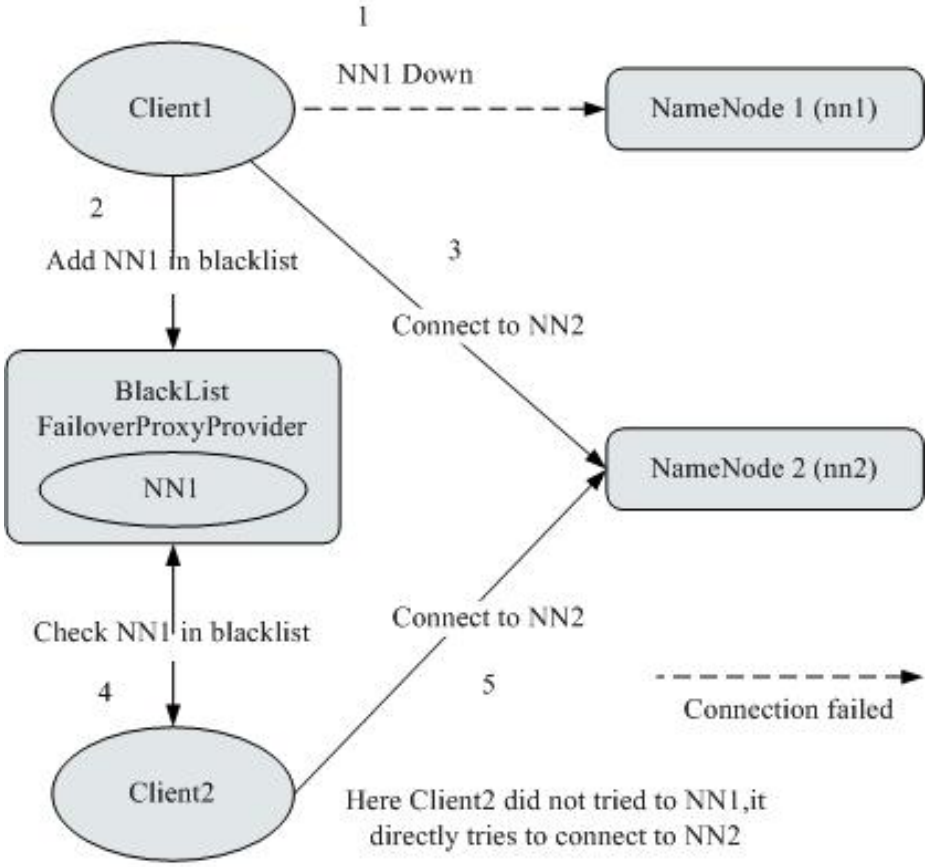
In the preceding cluster, if the active **nn1** cannot be accessed, client1 will retry the connection for 20 times. Then, a failover occurs, and client1 will connect to **nn2**. In the same way, other clients also connect to **nn2** when the failover occurs after retrying the connection to **nn1** for 20 times. Such process prolongs the fault recovery of NameNode.

In this case, the NameNode blacklisting adds **nn1** to the blacklist when client1 attempts to connect to the active **nn1** which is already faulty. Therefore, other clients will avoid trying to connect to **nn1** but choose **nn2** directly.

NOTE

If, at any time, all NameNodes are added to the blacklist, the content in the blacklist will be cleared, and the client attempts to connect to the NameNodes based on the initial NameNode list. If any fault occurs again, the NameNode is still added to the blacklist.

Figure 10-11 NameNode blacklisting working principle



Configuration Description

Go to the **All Configurations** page of HDFS and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 10-16 NameNode blacklisting parameters

Parameter	Description	Default Value
dfs.client.failover.proxy.provider.*[nameservice ID]*	Client Failover proxy provider class which creates the NameNode proxy using the authenticated protocol. Set this parameter to **org.apache.hadoop.hdfs.server.namenode.ha.BlackListingFailoverProxyProvider**. You can configure the observer NameNode to process read requests.	org.apache.hadoop.hdfs.server.namenode.ha.AdaptiveFailoverProxyProvider

10.13.5 Configuring Encrypted Channels

Scenario

Encrypted channel is an encryption protocol of remote procedure call (RPC) in HDFS. When a user invokes RPC, the user's login name will be transmitted to RPC through RPC head. Then RPC uses Simple Authentication and Security Layer (SASL) to determine an authorization protocol (Kerberos and DIGEST-MD5) to complete RPC authorization. When you deploy a security cluster, use a secure encrypted channel and configure the following parameters: For details about secure Hadoop RPC, visit the following website:

Versions earlier than MRS 3.2.0: **https://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-common/SecureMode.html#Data_Encryption_on_RPC**

MRS 3.2.0 or later: **https://hadoop.apache.org/docs/r3.3.1/hadoop-project-dist/hadoop-common/SecureMode.html#Data_Encryption_on_RPC**

Configuration Description

Go to the **All Configurations** page of HDFS and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 10-17 Parameter description

Parameter	Description	Default Value
hadoop.rpc.protection	**NOTICE** • The setting takes effect only after the service is restarted. Rolling restart is not supported. • After the setting, you need to download the client configuration again. Otherwise, the HDFS cannot provide the read and write services. Whether the RPC channels of each module in Hadoop are encrypted. The channels include: • RPC channels for clients to access HDFS • RPC channels between modules in HDFS, for example, RPC channels between DataNode and NameNode • RPC channels for clients to access Yarn • RPC channels between NodeManager and ResourceManager • RPC channels for Spark to access Yarn and HDFS • RPC channels for MapReduce to access Yarn and HDFS • RPC channels for HBase to access HDFS **NOTE** You can set this parameter on the HDFS component configuration page. The parameter setting takes effect globally, that is, the setting of whether the RPC channel is encrypted takes effect on all modules in Hadoop. There are three encryption modes. • **authentication**: This is the default value in normal mode. In this mode, data is directly transmitted without encryption after being authenticated. This mode ensures performance but has security risks. • **integrity**: Data is transmitted without encryption or authentication. To ensure data security, exercise caution when using this mode. • **privacy**: This is the default value in security mode, indicating that data is transmitted after authentication and encryption. This mode reduces the performance.	• Security mode: privacy • Normal mode: authentication

10.13.6 Configuring HDFS Hedged Read

This section is available for MRS 3.3.1 or later version only.

Scenario

In traditional HDFS, when a client requests to read data, it communicates with the NameNode to determine the DataNodes where the data block is and then connects to one node for data transmission. If the connected DataNode responds slowly or is faulty, the client must wait before attempting to obtain data from other replicas. There is a read latency. Enabling hedged read improves HDFS reliability in a high-latency network environment.

- Low read latency: The same data block is read from multiple data nodes at the same time.
- Adaptive to network changes: When the network is unstable or performance deteriorates on some nodes, the read efficiency of the client is improved.

Impact on the System

- Hedged read increases network loads and CPU usage because more connections and requests need to be processed. Enable this function based on the hardware and job conditions on the live network. For example, hedged read is enabled by default in a system that has three replicas. You need to set the component memory to at least three times the existing memory.
- When the disk I/O load is high (greater than 50% during peak hours), enabling hedged read may cause low disk performance deterioration.

Procedure

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **HDFS** and click the **Configurations** tab and then **All Configurations**.

Step 3 Search for **hdfs.hdfs-site.customized.configs** and add the custom parameters listed in the following table. Set the parameters based on the site requirements.

Parameter	Description	Value Range
dfs.client.hedged.read.threshold.millis	The number of milliseconds the client waits for the first byte of the first data block before deciding whether to start a hedged read	Greater than or equal to 0
dfs.client.hedged.read.threadpool.size	Size of the hedged read thread pool. If this parameter is set to **0**, the hedged read function is disabled.	Greater than or equal to 0

Step 4 Save the settings.

Step 5 On the **Instance** page of HDFS, select all DataNode instances, choose **More** > **Instance Rolling Restart**, and wait until the rolling restart is complete.

----End

10.13.7 Configuring Fine-Grained Locks of HDFS

Scenario

In FusionInsight Manager historical versions, HDFS uses a global lock for read-write and write-write operations. However, not all read-write and write-write operations cause resource competition. Therefore, the fine-grained lock feature (FGL) is introduced in this version. FGL splits the global lock based on directories and operation types and only operations that compete for resources use the same lock, which greatly improves the read and write performance.

For example, if client 1 and client 2 write data to irrelevant directories A and B at the same time, they do not compete for resources and do not hold the same lock.

NOTE

This function is available in MRS 3.5.0 and later versions.

Procedure

Step 1 Log in to Manager as an MRS cluster administrator, for example, **admin**.

Step 2 Choose **Cluster** > **Services** > **HDFS** and click the **Configurations** tab and then **All Configurations**.

Step 3 Search for and modify the **dfs.namenode.fgl.enable** parameter. For more information, see the following table.

Parameter	Description	Remarks
dfs.namenode.fgl.enable	Specifies whether to enable the fine-grained lock. This parameter can improve the HDFS read and write performance once it is enabled.	Default value: **false**

Step 4 Save the configuration.

Step 5 On the **Instance** page of HDFS, select the active and standby NameNode instances, choose **More** > **Instance Rolling Restart**, and wait until the rolling restart is complete.

----End

10.13.8 HDFS Auto Recovery from Cluster Power-off

Scenario

HDFS data is written to the OS cache before being written to disks. HDFS considers that data write is complete after the data is written into the OS cache. The OS needs to write cached data to disks. If the cluster is powered off, the cached data will be lost and HDFS losses blocks. If this happens during HDFS startup, HDFS enters the safe mode and cannot be automatically recovered.

To solve this problem, HDFS provides the following configuration parameters to recover in the event of cluster power-off. You need to adjust the parameters based your needs.

- If **dfs.datanode.synconclose** is **true**, the system considers that the write operation is complete only after the OS cache data is written to the disk. This prevents data loss caused by cluster power-off. However, this will demange HDFS write performance.
- **dfs.namenode.safemode.threshold-pct** indicates the maximum percentage of blocks reported by DataNodes. If this threshold is reached, NameNodes automatically exit the safe mode. If this threshold is too small, there can be a large number of replicas during cluster startup.

NOTE

This function is available in MRS 3.5.0 and later versions.

Procedure

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **HDFS** and click the **Configurations** tab and then **All Configurations**.

Step 3 Search for and set the following parameters as required.

Parameter	Description	Default Value
dfs.datanode.synconclose	If this parameter is set to **false**, block data will not be written into the disk immediately in the event of power outage or system restart during the process of storing files, which may result in data loss. If this parameter is set to **true**, data loss can be avoided in the event of power outage or system restart, but the performance deteriorates. Set this parameter based on the application scenario.	false
dfs.namenode.safemode.threshold-pct	Percentage of blocks that meet the minimum replication requirements defined by **dfs.namenode.replication.min**. Value range: 0 to 1.0 • When the value is less than or equal to 0, NameNode will exit the safe mode without waiting for any blocks. • When the value is greater than 1, NameNode permanently keeps in the safe mode.	0.999999

Step 4 Save the configuration.

Step 5 In the HDFS **Instances** tab, select all NameNode and DataNode instances, choose **More** > **Instance Rolling Restart**, verify the password, confirm the operation impact, and click **OK**. Wait until the rolling restart is complete.

----End

10.14 HDFS Performance Tuning

10.14.1 Improving Write Performance

Scenario

Improve the HDFS write performance by modifying the HDFS attributes.

Procedure

Navigation path for setting parameters:

On FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services** > **HDFS** > **Configurations** and select **All Configurations**. Enter a parameter name in the search box.

Table 10-18 Parameters for improving HDFS write performance

Parameter	Description	Default Value
dfs.datanode.drop.cache.behind.reads	Specifies whether to enable a DataNode to automatically clear all data in the cache after the data in the cache is transferred to the client. • **true**: The cached data is discarded. This parameter needs to be configured on the DataNode. You are advised to set it to **true** if data is repeatedly read only a few times, so that the cache can be used by other operations. • **false**: You are advised to set it to **false** if data is read repeatedly for many times to improve the read speed. **NOTE** This parameter is optional for improving write performance. You can configure it as needed.	false

Parameter	Description	Default Value
dfs.client-write-packet-size	Specifies the size of the client write packet. When the HDFS client writes data to the DataNode, the data will be accumulated until a packet is generated. Then, the packet is transmitted over the network. This parameter specifies the size (unit: byte) of the data packet to be transmitted, which can be specified by each job. In the 10-Gigabit network, you can increase the value of this parameter to enhance the transmission throughput.	262144

10.14.2 Improving Read Performance Using Client Metadata Cache

Scenario

Improve the HDFS read performance by using the client to cache the metadata for block locations.

NOTE

This function is recommended only for reading files that are not modified frequently. Because the data modification done on the server side by some other client is invisible to the cache client, which may cause the metadata obtained from the cache to be outdated.

Procedure

Navigation path for setting parameters:

On FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services** > **HDFS** > **Configurations**, select **All Configurations**, and enter the parameter name in the search box.

Table 10-19 Parameter configuration

Parameter	Description	Default Value
dfs.client.metadata.cache.enabled	Enables or disables the client to cache the metadata for block locations. Set this parameter to **true** and use it along with the **dfs.client.metadata.cache.pattern** parameter to enable the cache.	false

Parameter	Description	Default Value
dfs.client.metadata.cache.pattern	Indicates the regular expression pattern of the path of the file to be cached. Only the metadata for block locations of these files is cached until the metadata expires. This parameter is valid only when **dfs.client.metadata.cache.enabled** is set to **true**. Example: **/test.*** indicates that all files whose paths start with **/test** are read. **NOTE** • To ensure consistency, configure a specific mode to cache only files that are not frequently modified by other clients. • The regular expression pattern verifies only the path of the URI, but not the schema and authority in the case of the Fully Qualified path.	-
dfs.client.metadata.cache.expiry.sec	Indicates the duration for caching metadata. The cache entry becomes invalid after its caching time exceeds this duration. Even metadata that is frequently used during the caching process can become invalid. Time suffixes **s**/**m**/**h** can be used to indicate second, minute, and hour, respectively. **NOTE** If this parameter is set to **0s**, the cache function is disabled.	60s
dfs.client.metadata.cache.max.entries	Indicates the maximum number of non-expired data items that can be cached at a time.	65536

NOTE

Call *DFSClient#clearLocatedBlockCache()* to completely clear the client cache before it expires.

The sample usage is as follows:

```
FileSystem fs = FileSystem.get(conf);
DistributedFileSystem dfs = (DistributedFileSystem) fs;
DFSClient dfsClient = dfs.getClient();
dfsClient.clearLocatedBlockCache();
```


10.14.3 Improving the Connection Between the Client and NameNode Using Current Active Cache

Scenario

When HDFS is deployed in high availability (HA) mode with multiple NameNode instances, the HDFS client needs to connect to each NameNode in sequence to determine which is the active NameNode and use it for client operations.

Once the active NameNode is identified, its details can be cached and shared to all clients running on the client host. In this way, each new client first tries to load the details of the active Name Node from the cache and save the RPC call to the standby NameNode, which can help a lot in abnormal scenarios, for example, when the standby NameNode cannot be connected for a long time.

When a fault occurs and the other NameNode is switched to the active state, the cached details are updated to the information about the current active NameNode.

Procedure

Navigation path for setting parameters:

On FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services** > **HDFS** > **Configurations**, select **All Configurations**, and enter the parameter name in the search box.

Table 10-20 Configuration parameters

Parameter	Description	Default Value
dfs.client.failover.proxy.provider.[nameservice ID]	Client Failover proxy provider class which creates the NameNode proxy using the authenticated protocol. If this parameter is set to **org.apache.hadoop.hdfs.server.namenode.ha.BlackListingFailoverProxyProvider**, you can use the NameNode blacklist feature on the HDFS client. If this parameter is set to **org.apache.hadoop.hdfs.server.namenode.ha.ObserverReadProxyProvider**, you can configure the observer NameNode to process read requests.	org.apache.hadoop.hdfs.server.namenode.ha.AdaptiveFailoverProxyProvider
dfs.client.failover.activeinfo.share.flag	Specifies whether to enable the cache function and share the detailed information about the current active NameNode with other clients. Set it to **true** to enable the cache function.	false

Parameter	Description	Default Value
dfs.client.failover.activeinfo.share.path	Specifies the local directory for storing the shared files created by all clients in the host. If a cache area is to be shared by different users, the directory must have required permissions (for example, creating, reading, and writing cache files in the specified directory).	/tmp
dfs.client.failover.activeinfo.share.io.timeout.sec	(Optional) Used to control timeout. The cache file is locked when it is being read or written, and if the file cannot be locked within the specified time, the attempt to read or update the caches will be abandoned. The unit is second.	5

NOTE

The cache files created by the HDFS client are reused by other clients, and thus these files will not be deleted from the local system. If this function is disabled, you may need to manually clear the data.

10.14.4 Reducing the Probability of Abnormal Client Application Operation When the Network Is Not Stable

Scenario

Clients probably encounter running errors when the network is not stable. Users can adjust the following parameter values to improve the running efficiency.

Configuration Description

Go to the **All Configurations** page of HDFS and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 10-21 Parameter description

Parameter	Description	Default Value
ha.health-monitor.rpc-timeout.ms	Timeout interval during the NameNode health check performed by ZKFC. Increasing this value can prevent dual active NameNodes and reduce the probability of application running exceptions on clients. Unit: millisecond. Value range: 30,000 to 3,600,000	180,000

Parameter	Description	Default Value
ipc.client.connect.max.retries.on.timeouts	Number of retry times when the socket connection between a server and a client times out. Value range: 1 to 256	45
ipc.client.connect.timeout	Timeout interval of the socket connection between a client and a server. Increasing the value of this parameter increases the timeout interval for setting up a connection. Unit: millisecond. Value range: 1 to 3,600,000	20,000

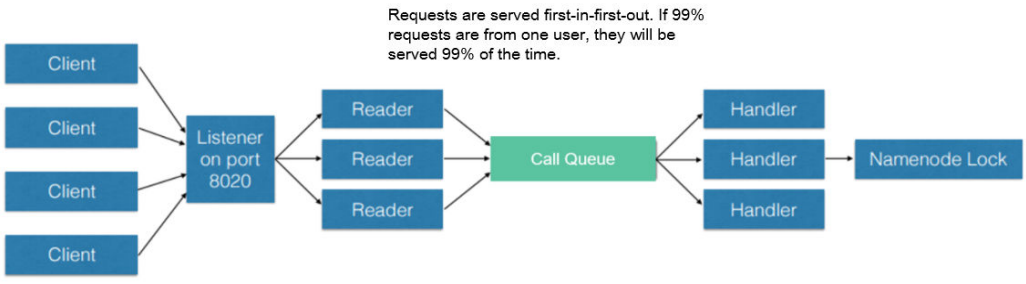
10.14.5 Optimizing HDFS NameNode RPC QoS

Scenarios

Several finished Hadoop clusters are faulty because the NameNode is overloaded and unresponsive.

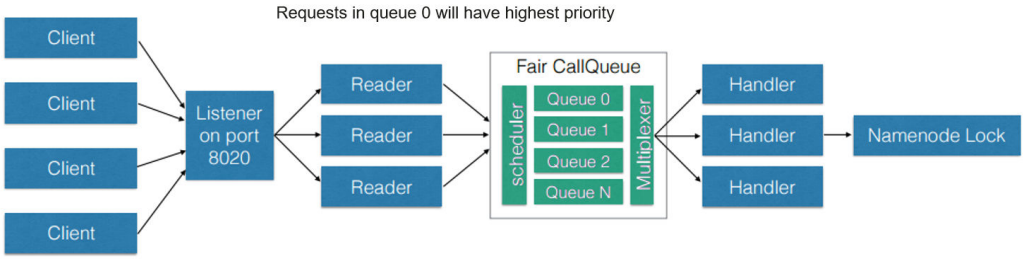
Such problem is caused by the initial design of Hadoop: In Hadoop, the NameNode functions as an independent part and in its namespace coordinates various HDFS operations, including obtaining the data block location, listing directories, and creating files. The NameNode receives HDFS operations, regards them as RPC calls, and places them in the FIFO call queue for read threads to process. Requests in FIFO call queue are served first-in first-out. However, users who perform more I/O operations are served more time than those performing fewer I/O operations. In this case, the FIFO is unfair and causes the delay.

Figure 10-12 NameNode request processing based on the FIFO call queue



The unfair problem and delaying mentioned before can be improved by replacing the FIFO queue with a new type of queue called FairCallQueue. In this way, FAIR queues assign incoming RPC calls to multiple queues based on the scale of the caller's call. The scheduling module tracks the latest calls and assigns a higher priority to users with a smaller number of calls.

Figure 10-13 NameNode request processing based on FAIRCallQueue



Configuration Description

- FairCallQueue ensures quality of service (QoS) by internally adjusting the order in which RPCs are invoked.

  This queue consists of the following parts:

  a. DecayRpcScheduler: used to provide priority values from 0 to N (the value 0 indicates the highest priority).

  b. Multi-level queues (located in the FairCallQueue): used to ensure that queues are invoked in order of priority.

  c. Multi-channel converters (provided with Weighted Round Robin Multiplexer): used to provide logic control for queue selection.

  After the FairCallQueue is configured, the control module determines the sub-queue to which the received invoking is allocated. The current scheduling module is DecayRpcScheduler, which only continuously tracks the priority numbers of various calls and periodically reduces these numbers.

  Go to the **All Configurations** page of HDFS and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

  Table 10-22 FairCallQueue parameters

Parameter	Description	Default Value
ipc.*<port>*.callqueue.impl	Specifies the queue implementation class. You need to run the **org.apache.hadoop.ipc.FairCallQueue** command to enable the QoS feature.	java.util.concurrent.LinkedBlockingQueue

- RPC BackOff

  Backoff is one of the FairCallQueue functions. It requires the client to retry operations (such as creating, deleting, and opening a file) after a period of time. When the backoff occurs, the RCP server throws RetriableException. The FairCallQueue performs backoff in either of the following cases:

  – The queue is full, that is, there are many client calls in the queue.

  – The queue response time is longer than the threshold time (specified by the **ipc.<port>.decay-scheduler.backoff.responsetime.thresholds** parameter).

Table 10-23 RPC Backoff configuration

Parameter	Description	Default Value
ipc.*<port>*.backoff.enable	Specifies whether to enable the backoff. When the current application contains a large number of user callings, the RPC request is blocked if the connection limit of the operating system is not reached. Alternatively, when the RPC or NameNode is heavily loaded, some explicit exceptions can be thrown back to the client based on certain policies. The client can understand these exceptions and perform exponential rollback, which is another implementation of the RetryInvocationHandler class.	false
ipc.*<port>*.decay-scheduler.backoff.responsetime.enable	Indicate whether to enable the backoff based on the average queue response time.	false
ipc.*<port>*.decay-scheduler.backoff.responsetime.thresholds	Configure the response time threshold for each queue. The response time threshold must match the number of priorities (the value of **ipc.<port> .faircallqueue.priority-levels**). Unit: millisecond	10000,20000,30000,40000

NOTE

- **<port>** indicates the RPC port configured on the NameNode.
- The backoff function based on the response time takes effect only when **ipc.<port> .backoff.enable** is set to **true**.

10.14.6 Optimizing HDFS DataNode RPC QoS

Scenario

When the speed at which the client writes data to the HDFS is greater than the disk bandwidth of the DataNode, the disk bandwidth is fully occupied. As a result, the DataNode does not respond. The client can back off only by canceling or restoring the channel, which results in write failures and unnecessary channel recovery operations.

Configuration

The new configuration parameter **dfs.pipeline.ecn** is introduced. When this configuration is enabled, the DataNode sends a signal from the write channel when the write channel is overloaded. The client may perform backoff based on the blocking signal to prevent the system from being overloaded. This configuration parameter is introduced to make the channel more stable and reduce unnecessary cancellation or recovery operations. After receiving the signal, the client backs off for a period of time (5,000 ms), and then adjusts the backoff time based on the related filter (the maximum backoff time is 50,000 ms).

Go to the **All Configurations** page of HDFS and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 10-24 DN ECN configuration

Parameter	Description	Default Value
dfs.pipeline.ecn	After configuration, the DataNode can send blocking notifications to the client.	false

10.14.7 Performing Concurrent Operations on HDFS Files

Scenario

Performing this operation can concurrently modify file and directory permissions and access control tools in a cluster.

Impact on the System

Performing concurrent file modification operations in a cluster has adverse impacts on the cluster performance. Therefore, you are advised to do so when the cluster is idle.

Prerequisites

- The HDFS client or clients including HDFS has been installed. For example, the installation directory is **/opt/client**.

- Service component users have been created by the MRS cluster administrator. In security mode, machine-machine users need to download the keytab file. A human-machine user needs to change the password upon the first login. (This operation is not required in normal mode.)

Procedure

Step 1 Log in to the node where the client is installed as the client installation user.

Step 2 Run the following command to go to the client installation directory:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 If the cluster is in security mode, the user executing the DistCp command must belong to the **supergroup** group and run the following command to perform user authentication. In normal mode, user authentication is not required.

kinit *Component service user*

Step 5 Increase the JVM size of the client to prevent out of memory (OOM). (32 GB is recommended for 100 million files.)

NOTE

The HDFS client exits abnormally and the error message "java.lang.OutOfMemoryError" is displayed after the HDFS client command is executed.

This problem occurs because the memory required for running the HDFS client exceeds the preset upper limit (128 MB by default). You can change the memory upper limit of the client by modifying **CLIENT_GC_OPTS** in *<Client installation path>*/**HDFS/component_env**. For example, if you want to set the upper limit to 1 GB, run the following command:

CLIENT_GC_OPTS="-Xmx1G"

After the modification, run the following command to make the modification take effect:

source <*Client installation path*>/**/bigdata_env**

Step 6 Run the concurrent commands shown in the following table.

Command	Description	Function
hdfs quickcmds [-t threadsNumber] [-p principal] [-k keytab] -setrep <rep> <path> ...	**threadsNumber** indicates the number of concurrent threads. The default value is the number of vCPUs of the local host. **principal** indicates the Kerberos user. **keytab** indicates the Keytab file. **rep** indicates the number of replicas. **path** indicates the HDFS directory.	Used to concurrently set the number of copies of all files in a directory.

Command	Description	Function
hdfs quickcmds [-t threadsNumber] [-p principal] [-k keytab] -chown [owner][:[group]] <path> ...	**threadsNumber** indicates the number of concurrent threads. The default value is the number of vCPUs of the local host. **principal** indicates the Kerberos user. **keytab** indicates the Keytab file. **owner** indicates the owner. **group** indicates the group to which the user belongs. **path** indicates the HDFS directory.	Used to concurrently set the owner group of all files in the directory.
hdfs quickcmds [-t threadsNumber] [-p principal] [-k keytab] -chmod <mode> <path> ...	**threadsNumber** indicates the number of concurrent threads. The default value is the number of vCPUs of the local host. **principal** indicates the Kerberos user. **keytab** indicates the Keytab file. **mode** indicates the permission (for example, 754). **path** indicates the HDFS directory.	Used to concurrently set permissions for all files in a directory.
hdfs quickcmds [-t threadsNumber] [-p principal] [-k keytab] -setfacl [{-b|-k} {-m|-x <acl_spec>} <path> ...]|[--set <acl_spec> <path> ...]	**threadsNumber** indicates the number of concurrent threads. The default value is the number of vCPUs of the local host. **principal** indicates the Kerberos user. **keytab** indicates the Keytab file. **acl_spec** indicates the ACL list separated by commas (,). **path** indicates the HDFS directory.	Used to concurrently set ACL information for all files in a directory.

----End

10.14.8 Configuring LZC Compression

Scenario

File compression can reduce the space occupied by stored files, and fasten data reading from disks and data transmission in the network. HDFS supports two default compression formats: Gzip and Snappy. For the new compression format Lempel-Ziv compression (LZC), this section describes its configuration procedures. This compression format enhances the Hadoop compression capability. For more information about Snappy, see https://code.google.com/p/snappy/.

Configuration Description

To make the LZC compression take effect, configure the following parameters in the **core-site.xml** file (for example, *Client installation path*/**HDFS/hadoop/etc/hadoop/**) of the client:

Table 10-25 Parameter Description

Parameter	Description	Default Value
io.compression.codecs	To make LZC take effect, the following values are added to the existing compression format list: **com.huawei.hadoop.datasight.io.compress.lzc.ZCodec** **NOTE** If more than one compression format is configured, use commas (,) to separate them.	org.apache.hadoop.io.compress.BZip2Codec,org.apache.hadoop.io.compress.DefaultCodec,org.apache.hadoop.io.compress.DeflateCodec,org.apache.hadoop.io.compress.Lz4Codec,org.apache.hadoop.io.compress.SnappyCodec,org.apache.hadoop.io.compress.GzipCodec,org.apache.hadoop.io.compress.ZStandardCodec,com.huawei.hadoop.datasight.io.compress.lzc.ZCodec
io.compression.codec.lzc.class	To make the LZC compression format take effect, use the default value. If you configure this parameter, set it to **com.huawei.hadoop.datasight.io.compress.lzc.ZCodec**.	com.huaweixxx.hadoop.datasight.io.compress.lzc.ZCodec

NOTE

1. LZC does not support FSImage and SequenceFile compression.
2. HDFS provides multiple compression algorithms, including Gzip, LZ4, Snappy, and Bzip2. The compression ratio and decompression speed of these compression algorithms are as follows:

   Compression ratio in descending order: Bzip2 > Gzip > LZ4 > Snappy

   Decompression speed in descending order: LZ4 > Snappy > Gzip > Bzip2
3. Application scenarios:
   - In scenarios where speed is required, for example, intermediate data storage of MapReduce tasks, LZ4 and Snappy are recommended. However, Snappy is recommended in scenarios requiring high reliability.
   - In scenarios where the compression ratio instead of compression speed is highly required, for example, cold data storage, Bzip2 or Gzip is recommended.
4. Except LZC, the preceding compression algorithms can be implemented using Native (C language). The compression and decompression efficiency is high. You are advised to use the compression algorithm supporting Native implementation based on service scenarios.

10.14.9 Asynchronously Deleting HDFS Data

Scenario

The HDFS asynchronous deletion feature is used to delete large directories. Deleting blocks asynchronously with traffic control can effectively shorten the continuous lock period.

NOTE

This function is available in MRS 3.5.0 or later only.

Procedure

Step 1 Go to the **All Configurations** page of HDFS by referring to **Modifying Cluster Service Configuration Parameters**.

Step 2 Enter the parameter name in the search box and **modify** the **HDFS service-level parameter** as you need.

Table 10-26 Parameters for configuring HDFS asynchronous deletion

Parameter	Description	Default Value
dfs.namenode.block.deletion.async	Whether to enable asynchronous block deletion.	true
dfs.namenode.block.deletion.lock.threshold.ms	Maximum duration for continuously lock a directory or file. After asynchronous block deletion is enabled, the specified maximum value is applied. Unit: millisecond	50

Parameter	Description	Default Value
dfs.namenode.block.deletion.lock.threshold.ms	Temporary unlock duration when the lock duration exceeds the maximum. Unit: millisecond	10

Step 3 Save the configuration.

Step 4 Click the **Instances** tab, select all instances whose configurations have expired, choose **More** > **Restart Instance**, and restart the instances as prompted.

----End

10.15 HDFS O&M Management

10.15.1 Configuring HDFS Parameters

Page Access

Go to the HDFS configurations page by referring to **Modifying Cluster Service Configuration Parameters**.

Parameter Description

Table 10-27 HDFS parameters

Parameter	Description	Default Value
fs.obs.security.provider	Implementation method of obtaining the key for accessing the OBS file system Value options are as follows: • **com.huawei.mrs.MrsObsCredentialsProvider**: obtains a credential through an MRS agency. • **com.obs.services.EcsObsCredentialsProvider**: obtains an AK/SK through the ECS service. • **com.obs.services.BasicObsCredentialsProvider**: uses the AK/SK information transferred to OBS. • **com.obs.services.EnvironmentVariableObsCredentialsProvider**: reads AK/SK information from environment variables.	com.huawei.mrs.MrsObsCredentialsProvider

10.15.2 Introduction to HDFS Logs

Log Description

Log path: The default path of HDFS logs is **/var/log/Bigdata/hdfs/***Role name*.

- NameNode: **/var/log/Bigdata/hdfs/nn** (run logs) and **/var/log/Bigdata/audit/hdfs/nn** (audit logs)
- DataNode: **/var/log/Bigdata/hdfs/dn** (run logs) and **/var/log/Bigdata/audit/hdfs/dn** (audit logs)
- ZKFC: **/var/log/Bigdata/hdfs/zkfc** (run logs) and **/var/log/Bigdata/audit/hdfs/zkfc** (audit logs)
- JournalNode: **/var/log/Bigdata/hdfs/jn** (run logs) and **/var/log/Bigdata/audit/hdfs/jn** (audit logs)
- Router: **/var/log/Bigdata/hdfs/router** (run logs) and **/var/log/Bigdata/audit/hdfs/router** (audit logs)
- HttpFS: **/var/log/Bigdata/hdfs/httpfs** (run logs) and **/var/log/Bigdata/audit/hdfs/httpfs** (audit logs)

Log archive rule: The automatic HDFS log compression function is enabled. By default, when the size of logs exceeds 100 MB, logs are automatically compressed into a log file named in the following format: *<Original log file name>-<yyyy-mm-dd_hh-mm-ss.[ID]>***.log.zip**. A maximum of 100 latest compressed files are reserved. The number of compressed files can be configured on Manager.

Table 10-28 HDFS log list

Type	Name	Description
Run log	hadoop-<SSH_USER>-<process_name>-<hostname>.log	HDFS system log, which records most of the logs generated when the HDFS system is running.
	hadoop-<SSH_USER>-<process_name>-<hostname>.out	Log that records the HDFS running environment information.
	hadoop.log	Log that records the operation of the Hadoop client.
	hdfs-period-check.log	Log that records scripts that are executed periodically, including automatic balancing, data migration, and JournalNode data synchronization detection.

Type	Name	Description
	<process_name>-<SSH_USER>-<DATE>-<PID>-gc.log	Garbage collection log file
	postinstallDetail.log	Work log before the HDFS service startup and after the installation.
	hdfs-service-check.log	Log that records whether the HDFS service starts successfully.
	hdfs-set-storage-policy.log	Log that records the HDFS data storage policies.
	cleanupDetail.log	Log that records the cleanup logs about the uninstallation of the HDFS service.
	prestartDetail.log	Log that records cluster operations before the HDFS service startup.
	hdfs-recover-fsimage.log	Recovery log of the NameNode metadata.
	datanode-disk-check.log	Log that records the disk status check during the cluster installation and use.
	hdfs-availability-check.log	Log that check whether the HDFS service is available.
	hdfs-backup-fsimage.log	Backup log of the NameNode metadata.
	startDetail.log	Detailed log that records the HDFS service startup.
	hdfs-blockplacement.log	Log that records the placement policy of HDFS blocks.
	upgradeDetail.log	Upgrade logs.
	hdfs-clean-acls-java.log	Log that records the clearing of deleted roles' ACL information by HDFS.

Type	Name	Description
	hdfs-haCheck.log	Run log that checks whether the NameNode in active or standby state has obtained scripts.
	<process_name>-jvmpause.log	Log that records JVM pauses during process running.
	hadoop-<SSH_USER>-balancer-<hostname>.log	Run log of HDFS automatic balancing.
	hadoop-<SSH_USER>-balancer-<hostname>.out	Log that records information of the environment where HDFS executes automatic balancing.
	hdfs-switch-namenode.log	Run log that records the HDFS active/standby switchover.
	hdfs-router-admin.log	Run log of the mount table management operation
	threadDump-<DATE>.log	Instance process stack log
Tomcat logs	hadoop-omm-host1.out, httpfs-catalina.<DATE>.log, httpfs-host-manager.<DATE>.log, httpfs-localhost.<DATE>.log, httpfs-manager.<DATE>.log, localhost_access_web_log.log	Tomcat run log
Audit log	hdfs-audit-<process_name>.log ranger-plugin-audit.log	Audit log that records the HDFS operations (such as creating, deleting, modifying and querying files).
	SecurityAuth.audit	HDFS security audit log.

Log Level

Table 10-29 lists the log levels supported by HDFS. The log levels include FATAL, ERROR, WARN, INFO, and DEBUG. Logs of which the levels are higher than or equal to the set level will be printed by programs. The higher the log level is set, the fewer the logs are recorded.

Table 10-29 Log levels

Level	Description
FATAL	Indicates the critical error information about system running.
ERROR	Indicates the error information about system running.
WARN	Indicates that the current event processing exists exceptions.
INFO	Indicates that the system and events are running properly.
DEBUG	Indicates the system and system debugging information.

To modify log levels, perform the following operations:

Step 1 Go to the **All Configurations** page of HDFS by referring to **Modifying Cluster Service Configuration Parameters**.

Step 2 On the left menu bar, select the log menu of the target role.

Step 3 Select a desired log level.

Step 4 Save the configuration. In the displayed dialog box, click **OK** to make the configurations take effect.

NOTE

The configurations take effect immediately without restarting the service.

----End

Log Formats

The following table lists the HDFS log formats.

Table 10-30 Log formats

Type	Format	Example
Run log	*<yyyy-MM-dd HH:mm:ss,SSS>*|*<Log level>*|*<Name of the thread that generates the log>*|*<Message in the log>*|*<Location where the log event occurs>*	2015-01-26 18:43:42,840 | INFO | IPC Server handler 40 on 8020 | Rolling edit logs | org.apache.hadoop.hdfs.server.namenode.FSEditLog.rollEditLog(FSEditLog.java:1096)

Type	Format	Example
Audit log	*<yyyy-MM-dd HH:mm:ss,SSS>*|*<Log level>*|*<Name of the thread that generates the log>*|*<Message in the log>*|*<Location where the log event occurs>*	2015-01-26 18:44:42,607 | INFO | IPC Server handler 32 on 8020 | allowed=true ugi=hbase (auth:SIMPLE) ip=/10.177.112.145 cmd=getfileinfo src=/hbase/WALs/hghoulaslx410,16020,1421743096083/hghoulaslx410%2C16020%2C1421743096083.1422268722795 dst=null perm=null | org.apache.hadoop.hdfs.server.namenode.FSNamesystem$DefaultAuditLogger.logAuditMessage(FSNamesystem.java:7950)

10.15.3 Planning HDFS Capacity

In HDFS, DataNode stores user files and directories as blocks, and file objects are generated on the NameNode to map each file, directory, and block on the DataNode.

The file objects on the NameNode require certain memory capacity. The memory consumption linearly increases as more file objects generated. The number of file objects on the NameNode increases and the objects consume more memory when the files and directories stored on the DataNode increase. In this case, the existing hardware may not meet the service requirement and the cluster is difficult to be scaled out.

Capacity planning of the HDFS that stores a large number of files is to plan the capacity specifications of the NameNode and DataNode and to set parameters according to the capacity plans.

Capacity Specifications

- NameNode capacity specifications

  Each file object on the NameNode corresponds to a file, directory, or block on the DataNode.

  A file uses at least one block. The default size of a block is **134,217,728**, that is, 128 MB, which can be set in the **dfs.blocksize** parameter. By default, a file whose size is less than 128 MB occupies only one block. If the file size is greater than 128 MB, the number of occupied blocks is the file size divided by 128 MB (Number of occupied blocks = File size/128). The directories do not occupy any blocks.

Based on **dfs.blocksize**, the number of file objects on the NameNode is calculated as follows:

Table 10-31 Number of NameNode file objects

Size of a File	Number of File Objects
< 128 MB	1 (File) + 1 (Block) = 2
> 128 MB (for example, 128 GB)	1 (File) + 1,024 (128 GB/128 MB = 1,024 blocks) = 1,025

The maximum number of file objects supported by the active and standby NameNodes is 300,000,000 (equivalent to 150,000,000 small files). **dfs.namenode.max.objects** specifies the number of file objects that can be generated in the system. The default value is **0**, which indicates that the number of generated file objects is not limited.

- DataNode capacity specifications

  In HDFS, blocks are stored on the DataNode as copies. The default number of copies is **3**, which can be set in the **dfs.replication** parameter.

  The number of blocks stored on all DataNode role instances in the cluster can be calculated based on the following formula: Number of HDFS blocks x 3 Average number of saved blocks = Number of HDFS blocks x 3/Number of DataNodes

Table 10-32 DataNode specifications

Item	Specifications
Maximum number of blocks supported by a DataNode instance	5,000,000
Maximum number of blocks supported by a disk on a DataNode instance	500,000
Minimum number of disks required when the number of blocks supported by a DataNode instance reaches the maximum	10

Table 10-33 Number of DataNodes

Number of HDFS Blocks	Minimum Number of DataNode Roles
10,000,000	10,000,000 *3/5,000,000 = 6
50,000,000	50,000,000 *3/5,000,000 = 30
100,000,000	100,000,000 *3/5,000,000 = 60

Setting Memory Parameters

- Configuration rules of the NameNode JVM parameter

  Default value of the NameNode JVM parameter **GC_OPTS**:

  -Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=128M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -Djdk.tls.ephemeralDHKeySize=3072 -Djdk.tls.rejectClientInitiatedRenegotiation=true -Djava.io.tmpdir=${Bigdata_tmp_dir}

  The number of NameNode files is proportional to the used memory size of the NameNode. When file objects change, you need to change **-Xms2G -Xmx4G -XX:NewSize=128M --XX:MaxNewSize=256M** in the default value. The following table lists the reference values.

  Table 10-34 NameNode JVM configuration

Number of File Objects	Reference Value
10,000,000	-Xms6G -Xmx6G -XX:NewSize=512M -XX:MaxNewSize=512M
20,000,000	-Xms12G -Xmx12G -XX:NewSize=1G -XX:MaxNewSize=1G
50,000,000	-Xms32G -Xmx32G -XX:NewSize=3G -XX:MaxNewSize=3G
100,000,000	-Xms64G -Xmx64G -XX:NewSize=6G -XX:MaxNewSize=6G
200,000,000	-Xms96G -Xmx96G -XX:NewSize=9G -XX:MaxNewSize=9G
300,000,000	-Xms164G -Xmx164G -XX:NewSize=12G -XX:MaxNewSize=12G

- Configuration rules of the DataNode JVM parameter

  Default value of the DataNode JVM parameter **GC_OPTS**:

  -Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=128M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -

XX:GCLogFileSize=1M -Djdk.tls.ephemeralDHKeySize=3072 -Djdk.tls.rejectClientInitiatedRenegotiation=true -Djava.io.tmpdir=${Bigdata_tmp_dir}

The average number of blocks stored in each DataNode instance in the cluster is: Number of HDFS blocks x 3/Number of DataNodes. If the average number of blocks changes, you need to change **-Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M** in the default value. The following table lists the reference values.

Table 10-35 DataNode JVM configuration

Average Number of Blocks in a DataNode Instance	Reference Value
2,000,000	-Xms6G -Xmx6G -XX:NewSize=512M -XX:MaxNewSize=512M
5,000,000	-Xms12G -Xmx12G -XX:NewSize=1G -XX:MaxNewSize=1G

Xmx specifies memory which corresponds to the threshold of the number of DataNode blocks, and each GB memory supports a maximum of 500,000 DataNode blocks. Set the memory as required.

Viewing the HDFS Capacity Status

- NameNode information

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** > **NameNode(Active)**, and click **Overview** to view information like the number of file objects, files, directories, and blocks in HDFS in **Summary** area.

- DataNode information

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** > **NameNode(Active)**, and click **DataNodes** to view the number of blocks on all DataNodes that report alarms.

- Alarm information

  Check whether the alarms whose IDs are 14007, 14008, and 14009 are generated and change the alarm thresholds as required.

10.15.4 Changing the DataNode Storage Directory

Scenario

If the storage directory defined by the HDFS DataNode is incorrect or the HDFS storage plan changes, the MRS cluster administrator needs to modify the DataNode storage directory on FusionInsight Manager to ensure smooth HDFS running. Changing the ZooKeeper storage directory includes the following scenarios:

- Change the storage directory of the DataNode role. In this way, the storage directories of all DataNode instances are changed.

- Change the storage directory of a single DataNode instance. In this way, only the storage directory of this instance is changed, and the storage directories of other instances remain the same.

Impact on the System

- The HDFS service needs to be stopped and restarted during the process of changing the storage directory of the DataNode role, and the cluster cannot provide services before it is completely started.
- The DataNode instance needs to stopped and restarted during the process of changing the storage directory of the instance, and the instance at this node cannot provide services before it is started.
- The directory for storing service parameter configurations must also be updated.

Prerequisites

- New disks have been prepared and installed on each data node, and the disks are formatted.
- New directories have been planned for storing data in the original directories.
- The HDFS client has been installed.
- The service user **hdfs** is available.
- When changing the storage directory of a single DataNode instance, ensure that the number of active DataNode instances is greater than the value of **dfs.replication**.

Procedure

Check the environment.

Step 1 Log in to the server where the HDFS client is installed as user **root**, and run the following command to configure environment variables:

source *Installation directory of the HDFS client*/**bigdata_env**

Step 2 If the cluster is in security mode, run the following command to authenticate the user:

kinit hdfs

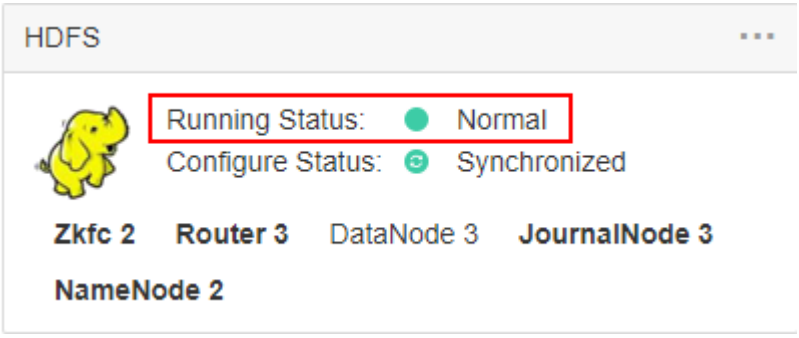
Step 3 Run the following command on the HDFS client to check whether all directories and files in the HDFS root directory are normal:

hdfs fsck /

Check the fsck command output.

- If the following information is displayed, no file is lost or damaged. Go to **Step 4**.
  ```
  The filesystem under path '/' is HEALTHY
  ```
- If other information is displayed, some files are lost or damaged. Go to **Step 5**.

Step 4 Log in to FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services**, and check whether **Running Status** of HDFS is **Normal**.



- If yes, go to **Step 6**.
- If no, the HDFS status is unhealthy. Go to **Step 5**.

Step 5 Rectify the HDFS fault.. The task is complete.

Step 6 Determine whether to change the storage directory of the DataNode role or that of a single DataNode instance:

- To change the storage directory of the DataNode role, go to **Step 7**.
- To change the storage directory of a single DataNode instance, go to **Step 12**.

Changing the storage directory of the DataNode role

Step 7 Choose **Cluster** > *Name of the desired cluster* > **Services** > **HDFS** > **Stop Instance** to stop the HDFS service.

Step 8 Log in to each data node where the HDFS service is installed as user **root** and perform the following operations:

1. Create a target directory (**data1** and **data2** are original directories in the cluster).

   For example, to create a target directory **${BIGDATA_DATA_HOME}/hadoop/data3/dn**, run the following command:

   mkdir -p ${BIGDATA_DATA_HOME}/hadoop/data3/dn

2. Mount the target directory to the new disk. For example, mount **${BIGDATA_DATA_HOME}/hadoop/data3** to the new disk.
3. Modify permissions on the new directory.

   For example, to create a target directory **${BIGDATA_DATA_HOME}/hadoop/data3/dn**, run the following commands:

   chmod 700 ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R and **chown omm:wheel ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R**

4. Copy the data to the target directory.

   For example, if the old directory is **${BIGDATA_DATA_HOME}/hadoop/data1/dn** and the target directory is **${BIGDATA_DATA_HOME}/hadoop/data3/dn**, run the following command:

   cp -af ${BIGDATA_DATA_HOME}/hadoop/data1/dn/* ${BIGDATA_DATA_HOME}/hadoop/data3/dn

Step 9 On FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services** > **HDFS** > **Configurations** > **All Configurations** to go to the HDFS service configuration page.

Change the value of **dfs.datanode.data.dir** from the default value **%{@auto.detect.datapart.dn}** to the new target directory, for example, **${BIGDATA_DATA_HOME}/hadoop/data3/dn**.

For example, the original data storage directories are **/srv/BigData/hadoop/data1**, **/srv/BigData/hadoop/data2**. To migrate data from the **/srv/BigData/hadoop/data1** directory to the newly created **/srv/BigData/hadoop/data3** directory, replace the whole parameter with **/srv/BigData/hadoop/data2, /srv/BigData/hadoop/data3**. Separate multiple storage directories with commas (,). In this example, changed directories are **/srv/BigData/hadoop/data2**, **/srv/BigData/hadoop/data3**.

Step 10 Click **Save**. Choose **Cluster** > *Name of the desired cluster* > **Services**. On the page that is displayed, start the services that have been stopped.

Step 11 After the HDFS is started, run the following command on the HDFS client to check whether all directories and files in the HDFS root directory are correctly copied:

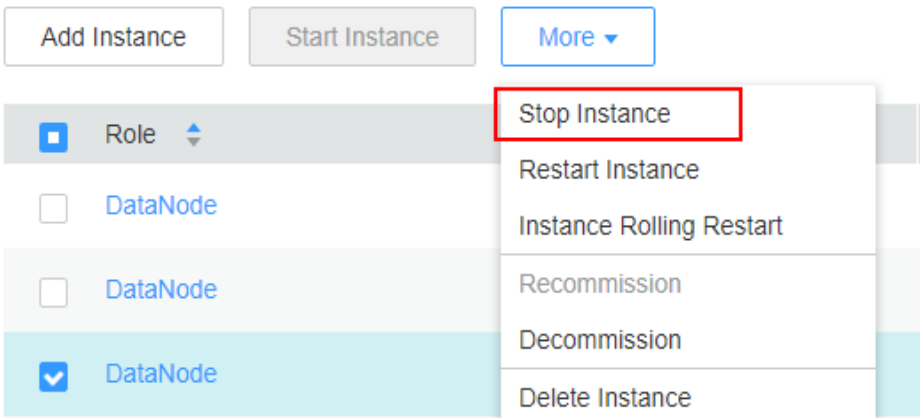
hdfs fsck /

Check the fsck command output.

- If the following information is displayed, no file is lost or damaged, and data replication is successful. No further action is required.
  ```
  The filesystem under path '/' is HEALTHY
  ```
- If other information is displayed, some files are lost or damaged. In this case, check whether **8.4** is correct and run the **hdfs fsck** *Name of the damaged file* **-delete** command.

Changing the storage directory of a single DataNode instance

Step 12 Choose **Cluster** > *Name of the desired cluster* > **Services** > **HDFS** > **Instance**. Select the HDFS instance whose storage directory needs to be modified, and choose **More** > **Stop Instance**.



Step 13 Log in to the DataNode node as user **root**, and perform the following operations:

1. Create a target directory.

   For example, to create a target directory **${BIGDATA_DATA_HOME}/hadoop/data3/dn**, run the following command:

   mkdir -p ${BIGDATA_DATA_HOME}/hadoop/data3/dn
2. Mount the target directory to the new disk.

For example, mount **${BIGDATA_DATA_HOME}/hadoop/data3** to the new disk.

3. Modify permissions on the new directory.

   For example, to create a target directory **${BIGDATA_DATA_HOME}/hadoop/data3/dn**, run the following commands:

   chmod 700 ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R and **chown omm:wheel ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R**

4. Copy the data to the target directory.

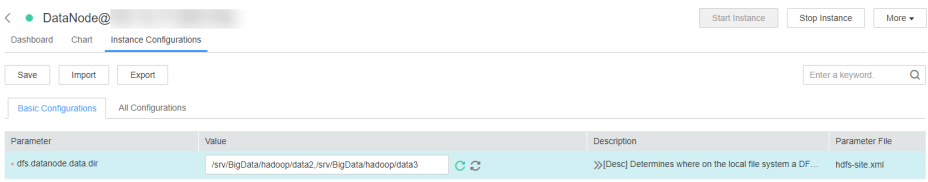
   For example, if the old directory is **${BIGDATA_DATA_HOME}/hadoop/data1/dn** and the target directory is **${BIGDATA_DATA_HOME}/hadoop/data3/dn**, run the following command:

   cp -af ${BIGDATA_DATA_HOME}/hadoop/data1/dn/* ${BIGDATA_DATA_HOME}/hadoop/data3/dn

Step 14 On FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Service** > **HDFS** > **Instance**. Click the specified DataNode instance and go to the **Configurations** page.

Change the value of **dfs.datanode.data.dir** from the default value **%{@auto.detect.datapart.dn}** to the new target directory, for example, **${BIGDATA_DATA_HOME}/hadoop/data3/dn**.

For example, the original data storage directories are **/srv/BigData/hadoop/data1,/srv/BigData/hadoop/data2**. To migrate data from the **/srv/BigData/hadoop/data1** directory to the newly created **/srv/BigData/hadoop/data3** directory, replace the whole parameter with **/srv/BigData/hadoop/data2,/srv/BigData/hadoop/data3**.



Step 15 Click **Save**, and then click **OK**.

Operation succeeded is displayed. click **Finish**.

Step 16 Choose **More** > **Restart Instance** to restart the DataNode instance.

----End

10.15.5 Configuring the Damaged Disk Volume

Scenario

In the open source version, if multiple data storage volumes are configured for a DataNode, the DataNode stops providing services by default if one of the volumes is damaged. You can change the value of **dfs.datanode.failed.volumes.tolerated** to specify the number of damaged disk volumes that are allowed. If the number of damaged volumes does not exceed the threshold, DataNode continues to provide services.

Configuration Description

Navigation path for setting parameters:

Go to the **All Configurations** page of HDFS and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 10-36 Parameter description

Parameter	Description	Default Value
dfs.datanode.failed.volumes.tolerated	Specifies the number of damaged volumes that are allowed before the DataNode stops providing services. By default, there must be at least one valid volume. The value **-1** indicates that the minimum value of a valid volume is **1**. The value greater than or equal to **0** indicates the number of damaged volumes that are allowed.	–1

10.15.6 Setting the Maximum Lifetime and Renewal Interval of a Token

Scenario

In security mode, users can flexibly set the maximum token lifetime and token renewal interval in HDFS based on cluster requirements.

Configuration Description

Navigation path for setting parameters:

Go to the **All Configurations** page of HDFS and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 10-37 Parameter description

Parameter	Description	Default Value
dfs.namenode.delegation.token.max-lifetime	This parameter is a server parameter. It specifies the maximum lifetime of a token. Unit: milliseconds. Value range: 10,000 to 10,000,000,000,000	604,800,000
dfs.namenode.delegation.token.renew-interval	This parameter is a server parameter. It specifies the maximum lifetime to renew a token. Unit: milliseconds. Value range: 10,000 to 10,000,000,000,000	86,400,000

10.15.7 Running the DistCp Command

Scenario

DistCp is a tool used to perform large-amount data replication between clusters or in a cluster. It uses MapReduce tasks to implement distributed copy of a large amount of data.

Prerequisites

- The Yarn client or a client that contains Yarn has been installed. For example, the installation directory is **/opt/client**.
- Service users of each component are created by the MRS cluster administrator based on service requirements. In security mode, machine-machine users need to download the keytab file. A human-machine user must change the password upon the first login. (Not involved in normal mode)
- To copy data between clusters, you need to enable the inter-cluster data copy function on both clusters.

Procedure

Step 1 Log in to the node where the client is installed.

Step 2 Run the following command to go to the client installation directory:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 If the cluster is in security mode, the user group to which the user executing the DistCp command belongs must be **supergroup** and the user run the following command to perform user authentication. In normal mode, user authentication is not required.

kinit *Component service user*

Step 5 Run the DistCp command. The following provides an example:

hadoop distcp hdfs://hacluster/source hdfs://hacluster/target

----End

Common Usage of DistCp

1. The following is an example of the commonest usage of DistCp:
   ```
   hadoop distcp -numListstatusThreads 40 -update -delete -prbugpaxtq hdfs://cluster1/source hdfs://cluster2/target
   ```


NOTE

In the preceding command:

- **-numListstatusThreads** specifies the number of threads for creating the list of 40 copied files.
- **-update -delete** specifies that files at the source location and the target location are synchronized, and that files with excessive target locations are deleted. If you need to copy files incrementally, delete **-delete**.
- If **-prbugpaxtq** and **-update** are used, it indicates that the status information of the copied file is also updated.
- **hdfs://cluster1/source** indicates the source location, and **hdfs://cluster2/target** indicates the target location.

2. The following is an example of data copy between clusters:

```
hadoop distcp hdfs://cluster1/foo/bar hdfs://cluster2/bar/foo
```

NOTE

The network between cluster1 and cluster2 must be reachable, and the two clusters must use the same HDFS version or compatible HDFS versions.

3. The following are multiple examples of data copy in a source directory:

```
hadoop distcp hdfs://cluster1/foo/a \
hdfs://cluster1/foo/b \
hdfs://cluster2/bar/foo
```

The preceding command is used to copy the folders a and b of cluster1 to the **/bar/foo** directory of cluster2. The effect is equivalent to that of the following commands:

```
hadoop distcp -f hdfs://cluster1/srclist \
hdfs://cluster2/bar/foo
```

The content of **srclist** is as follows. Before running the DistCp command, upload the **srclist** file to HDFS.

```
hdfs://cluster1/foo/a
hdfs://cluster1/foo/b
```

4. **-update** indicates that a to-be-copied file does not exist in the target location, or the content of the copied file in the target location is updated; and **-overwrite** is used to overwrite existing files in the target location.

The following is an example of the difference between no option and any one of the two options (either **update** or **overwrite**) that is added:

Assume that the structure of a file at the source location is as follows:

```
hdfs://cluster1/source/first/1
hdfs://cluster1/source/first/2
hdfs://cluster1/source/second/10
hdfs://cluster1/source/second/20
```

Commands without options are as follows:

```
hadoop distcp hdfs://cluster1/source/first  hdfs://cluster1/source/second  hdfs://cluster2/target
```

By default, the preceding command creates the **first** and **second** folders at the target location. Therefore, the copy results are as follows:

```
hdfs://cluster2/target/first/1
hdfs://cluster2/target/first/2
hdfs://cluster2/target/second/10
hdfs://cluster2/target/second/20
```

The command with any one of the two options (for example, **update**) is as follows:

```
hadoop distcp -update hdfs://cluster1/source/first  hdfs://cluster1/source/second  hdfs://cluster2/target
```


The preceding command copies only the content at the source location to the target location. Therefore, the copy results are as follows:

```
hdfs://cluster2/target/1
hdfs://cluster2/target/2
hdfs://cluster2/target/10
hdfs://cluster2/target/20
```

NOTE

- If files with the same name exist in multiple source locations, the DistCp command fails.
- If neither **update** nor **overwrite** is used and the file to be copied already exists in the target location, the file will be skipped.
- When **update** is used, if the file to be copied already exists in the target location but the file content is different, the file content in the target location is updated.
- When **overwrite** is used, if the file to be copied already exists in the target location, the file in the target location is still overwritten.

5. The following table describes other command options:

Table 10-38 Other command options

Option	Description
-p[rbugpcaxtq]	When **-update** is also used, the status information of a copied file is updated even if the content of the copied file is not updated. **r**: number of copies **b**: size of a block **u**: user to which the files belong **g**: user group to which the user belongs **p**: permission **c**: check and type **a**: access control **t**: timestamp **q**: quota information
-i	Failures ignored during copying
-log <logdir>	Path of the specified log
-v	Additional information in the specified log
-m <num_maps>	Maximum number of concurrent copy tasks that can be executed at the same time
-numListstatusThreads	Number of threads for constituting the list of copied files. This option increases the running speed of DistCp.
-overwrite	File at the target location that is to be overwritten

Option	Description
-update	A file at the target location is updated if the size and check of a file at the source location are different from those of the file at the target location.
-append	When **-update** is also used, the content of the file at the source location is added to the file at the target location.
-f <urilist_uri>	Content of the **<urilist_uri>** file is used as the file list to be copied.
-filters	A local file is specified whose content contains multiple regular expressions. If the file to be copied matches a regular expression, the file is not copied.
-async	The **distcp** command is run asynchronously.
-atomic {-tmp <tmp_dir>}	An atomic copy can be performed. You can add a temporary directory during copying.
-bandwidth	The transmission bandwidth of each copy task. Unit: MB/s.
-delete	The files that exist in the target location is deleted but do not exist in the source location. This option is usually used with **-update**, and indicates that files at the source location are synchronized with those at the target location and the redundant files at the target location are deleted.
-diff <oldSnapshot> <newSnapshot>	The differences between the old and new versions are copied to a file in the old version at the target location.
-skipcrccheck	Whether to skip the cyclic redundancy check (CRC) between the source file and the target file.
-strategy {dynamic| uniformsize}	The policy for copying a task. The default policy is **uniformsize**, that is, each copy task copies the same number of bytes.

FAQs of DistCp

1. When you run the DistCp command, if the content of some copied files is large, you are advised to change the timeout period of MapReduce that executes the copy task. It can be implemented by specifying the **mapreduce.task.timeout** in the DistCp command. For example, run the following command to change the timeout to 30 minutes:

   ```
   hadoop distcp -Dmapreduce.task.timeout=1800000 hdfs://cluster1/source hdfs://cluster2/target
   ```

   Or, you can also use **filters** to exclude the large files out of the copy process. The command example is as follows:

   ```
   hadoop distcp -filters /opt/client/filterfile hdfs://cluster1/source hdfs://cluster2/target
   ```


In the preceding command, *filterfile* indicates a local file, which contains multiple expressions used to match the path of a file that is not copied. The following is an example:

```
.*excludeFile1.*
.*excludeFile2.*
```

2. If the DistCp command unexpectedly quits, the error message "java.lang.OutOfMemoryError" is displayed.

   This is because the memory required for running the copy command exceeds the preset memory limit (default value: 128 MB). You can change the memory upper limit of the client by modifying **CLIENT_GC_OPTS** in *<Client installation path>***/HDFS/component_env**. For example, if you want to set the memory upper limit to 1 GB, refer to the following configuration:

   ```
   CLIENT_GC_OPTS="-Xmx1G"
   ```

   After the modification, run the following command to make the modification take effect:

   source {*Client installation path*}**/bigdata_env**

3. When the dynamic policy is used to run the DistCp command, the command exits unexpectedly and the error message "Too many chunks created with splitRatio" is displayed.

   The cause of this problem is that the value of **distcp.dynamic.max.chunks.tolerable** (default value: 20,000) is less than the value of **distcp.dynamic.split.ratio** (default value: 2) multiplied by the number of Maps. This problem occurs when the number of Maps exceeds 10,000. You can use the **-m** parameter to reduce the number of Maps to less than 10,000.

   ```
   hadoop distcp -strategy dynamic -m 9500 hdfs://cluster1/source hdfs://cluster2/target
   ```

   Alternatively, you can use the **-D** parameter to set **distcp.dynamic.max.chunks.tolerable** to a large value.

   ```
   hadoop distcp -Ddistcp.dynamic.max.chunks.tolerable=30000 -strategy dynamic hdfs://cluster1/source hdfs://cluster2/target
   ```

10.15.8 Configuring NFS

Scenario

Before deploying a cluster, you can deploy a Network File System (NFS) server based on requirements to store NameNode metadata to enhance data reliability.

If the NFS server has been deployed and NFS services are configured, you can follow operations in this section to configure NFS on the cluster. These operations are optional.

Procedure

Step 1 Check the permission of the shared NFS directories on the NFS server to ensure that the server can access NameNode in the MRS cluster.

Step 2 Log in to the active NameNode as user **root**.

Step 3 Run the following commands to create a directory and assign it write permissions:

mkdir ${BIGDATA_DATA_HOME}/namenode-nfs

chown omm:wheel ${BIGDATA_DATA_HOME}/namenode-nfs

chmod 750 ${BIGDATA_DATA_HOME}/namenode-nfs

Step 4 Run the following command to mount the NFS to the active NameNode:

mount -t nfs -o rsize=8192,wsize=8192,soft,nolock,timeo=3,intr *IP address of the NFS server*:*Shared directory* **${BIGDATA_DATA_HOME}/namenode-nfs**

For example, if the IP address of the NFS server is **192.168.0.11** and the shared directory is **/opt/Hadoop/NameNode**, run the following command:

mount -t nfs -o rsize=8192,wsize=8192,soft,nolock,timeo=3,intr 192.168.0.11:/opt/Hadoop/NameNode ${BIGDATA_DATA_HOME}/namenode-nfs

Step 5 Perform **Step 2** to **Step 4** on the standby NameNode.

NOTE

The names of the shared directories (for example, **/opt/Hadoop/NameNode**) created on the NFS server by the active and standby NameNodes must be different.

Step 6 Log in to FusionInsight Manager, and choose **Cluster** > *Name of the desired cluster* > **Service** > **HDFS** > **Configuration** > **All Configurations**.

Step 7 In the search box, search for **dfs.namenode.name.dir**, add **${BIGDATA_DATA_HOME}/namenode-nfs** to **Value**, and click **Save**. Separate paths with commas (,).

Step 8 Click **OK**. On the **Dashboard** tab page, choose **More** > **Restart Service** to restart the service.

----End

10.16 Common commands of the HDFS client

Using the HDFS Client

Step 1 Install a client. For details, see **Installing a Client**.

Step 2 Log in to the node where the client is installed as the client installation user.

Step 3 Run the following command to go to the client installation directory:

cd /opt/client

Step 4 Run the following command to configure environment variables:

source bigdata_env

Step 5 If the cluster is in security mode, run the following command to authenticate the user. In normal mode, user authentication is not required.

kinit *Component service user*

Step 6 Run the HDFS Shell command. Example:

hdfs dfs -ls /

----End

Common HDFS Client Commands

The following table lists common HDFS client commands.

For more commands, see **https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/CommandsManual.html#User_Commands**.

Table 10-39 Common HDFS client commands

Command	Description	Example
hdfs dfs -mkdir *Folder name*	Used to create a folder.	**hdfs dfs -mkdir /tmp/mydir**
hdfs dfs -ls *Folder name*	Used to view a folder.	**hdfs dfs -ls /tmp**
hdfs dfs -put *Local file on the client node Specified HDFS path*	Used to upload a local file to a specified HDFS path.	**hdfs dfs -put /opt/test.txt /tmp** Upload the **/opt/test.txt** file on the client node to the **/tmp** directory of HDFS.
hdfs dfs -get *Specified file on HDFS Specified path on the client node*	Used to download the HDFS file to the specified local path.	**hdfs dfs -get /tmp/test.txt /opt/** Download the **/tmp/test.txt** file on HDFS to the **/opt** path on the client node.
hdfs dfs -rm -r -f *Specified folder on HDFS*	Used to delete a folder.	**hdfs dfs -rm -r -f /tmp/mydir**
hdfs dfs -chmod *Permission parameter File directory*	Used to configure the HDFS directory permission for a user.	**hdfs dfs -chmod 700 /tmp/test**

10.17 Common Issues About HDFS

10.17.1 Why Does the Distcp Command Fail in the Secure Cluster, Causing an Exception?

Question

Why distcp command fails in the secure cluster with the following error displayed?

Client side exception


```
Invalid arguments: Unexpected end of file from server
```

Server side exception

```
javax.net.ssl.SSLException: Unrecognized SSL message, plaintext connection?
```

Answer

The preceding error may occur if **webhdfs://** is used in the distcp command. The reason is that the big data cluster uses the HTTPS mechanism, that is, **dfs.http.policy** is set to **HTTPS_ONLY** in **core-site.xml** file. To avoid the error, replace **webhdfs://** with **swebhdfs://** in the file.

For example:

./hadoop distcp swebhdfs://IP:PORT/testfile hdfs://IP:PORT/testfile1

10.17.2 When Does a Balance Process in HDFS, Shut Down and Fail to be Executed Again?

Question

After I start a Balance process in HDFS, the process is shut down abnormally. If I attempt to execute the Balance process again, it fails again.

Answer

After a Balance process is executed in HDFS, another Balance process can be executed only after the **/system/balancer.id** file is automatically released.

However, if a Balance process is shut down abnormally, the **/system/balancer.id** has not been released when the Balance is executed again, which triggers the **append /system/balancer.id** operation.

- If the time spent on releasing the **/system/balancer.id** file exceeds the soft-limit lease period 60 seconds, executing the Balance process again triggers the append operation, which preempts the lease. The last block is in construction or under recovery status, which triggers the block recovery operation. The **/system/balancer.id** file cannot be closed until the block recovery completes. Therefore, the append operation fails.

  After the **append /system/balancer.id** operation fails, the exception message **RecoveryInProgressException** is displayed.

  ```
  org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.protocol.RecoveryInProgressException): Failed to APPEND_FILE /system/balancer.id for DFSClient because lease recovery is in progress. Try again later.
  ```

- If the time spent on releasing the **/system/balancer.id** file is within 60 seconds, the original client continues to own the lease and the exception AlreadyBeingCreatedException occurs and null is returned to the client. The following exception message is displayed on the client:

  ```
  java.io.IOException: Cannot create any NameNode Connectors.. Exiting...
  ```

Either of the following methods can be used to solve the problem:

- Execute the Balance process again after the hard-limit lease period expires for 1 hour, when the original client has released the lease.

- Delete the **/system/balancer.id** file before executing the Balance process again.

10.17.3 "This page can't be displayed" Is Displayed When Internet Explorer Fails to Access the Native HDFS UI

Question

Occasionally, nternet Explorer 9, Explorer 10, or Explorer 11 fails to access the native HDFS UI.

Symptom

Internet Explorer 9, Explorer 10, or Explorer 11 fails to access the native HDFS UI, as shown in the following figure.

Cause

Some Internet Explorer 9, Explorer 10, or Explorer 11versions fail to handle SSL handshake issues, causing access failure.

Solution

Refresh the page.

10.17.4 HDFS WebUI Cannot Properly Update Information About Damaged Data

Question

1. When errors occur in the **dfs.datanode.data.dir** directory of DataNode due to the permission or disk damage, HDFS WebUI does not display information about damaged data.
2. After errors are restored, HDFS WebUI does not timely remove related information about damaged data.

Answer

1. DataNode checks whether the disk is normal only when errors occur in file operations. Therefore, only when a data damage is detected and the error is reported to NameNode, NameNode displays information about the damaged data on HDFS WebUI.
2. After errors are fixed, you need to restart DataNode. During restarting DataNode, all data states are checked and damaged data information is uploaded to NameNode. Therefore, after errors are fixed, damaged data information is not displayed on the HDFS WebUI only by restarting DataNode.

10.17.5 The HDFS Client Is Unresponsive When the NameNode Is Overloaded for a Long Time

Question

When the NameNode node is overloaded (100% of the CPU is occupied), the NameNode is unresponsive. The HDFS clients that are connected to the overloaded NameNode fail to run properly. However, the HDFS clients that are newly connected to the NameNode will be switched to a backup NameNode and run properly.

Answer

The default configuration must be used (as described in **Table 10-40**) when the error preceding described occurs: the **keep alive** mechanism is enabled for the RPC connection between the HDFS client and the NameNode. The **keep alive** mechanism will keep the HDFS client waiting for the response from server and prevent the connection from being out timed, causing the unresponsiveness of the HDFS client.

Perform the following operations to the unresponsive HDFS client:

- Leave the HDFS client waiting. Once the CPU usage of the node where NameNode locates drops, the NameNode will obtain CPU resources and the HDFS client will receive a response.
- If you do not want to leave the HDFS client running, restart the application where the HDFS client locates to reconnect the HDFS client to another idle NameNode.

Procedure:

Configure the following parameters in the *client installation path*/**HDFS/hadoop/etc/hadoop/core-site.xml** file on the client.

Table 10-40 Parameter description

Parameter	Description	Default Value
ipc.client.ping	If the **ipc.client.ping** parameter is configured to **true**, the HDFS client will wait for the response from the server and periodically send the **ping** message to avoid disconnection caused by **tcp timeout**. If the **ipc.client.ping** parameter is configured to **false**, the HDFS client will set the value of **ipc.ping.interval** as the timeout time. If no response is received within that time, timeout occurs. To avoid the unresponsiveness of HDFS when the NameNode is overloaded for a long time, you are advised to set the parameter to **false**.	true

Parameter	Description	Default Value
ipc.ping.interval	If the value of **ipc.client.ping** is **true**, **ipc.ping.interval** indicates the interval between sending the ping messages. If the value of **ipc.client.ping** is **false**, **ipc.ping.interval** indicates the timeout time for connection. To avoid the unresponsiveness of HDFS when the NameNode is overloaded for a long time, you are advised to set the parameter to a large value, for example **900000** (unit ms) to avoid timeout when the server is busy.	60000

10.17.6 Why are There Two Standby NameNodes After the active NameNode Is Restarted?

Question

Why are there two standby NameNodes after the active NameNode is restarted?

When this problem occurs, check the ZooKeeper and ZooKeeper FC logs. You can find that the sessions used for the communication between the ZooKeeper server and client (ZKFC) are inconsistent. The session ID of the ZooKeeper server is **0x164cb2b3e4b36ae4**, and the session ID of the ZooKeeper FC is **0x144cb2b3e4b36ae4**. Such inconsistency means that the data interaction between the ZooKeeper server and ZKFC fails.

Content of the ZooKeeper log is as follows:

```
2015-04-15 21:24:54,257 | INFO | CommitProcessor:22 | Established session 0x164cb2b3e4b36ae4 with negotiated timeout 45000 for client /192.168.0.117:44586 | org.apache.zookeeper.server.ZooKeeperServer.finishSessionInit(ZooKeeperServer.java:623)
2015-04-15 21:24:54,261 | INFO | NIOServerCxn.Factory:192-168-0-114/192.168.0.114:2181 | Successfully authenticated client: authenticationID=hdfs/hadoop@<System domain name>; authorizationID=hdfs/hadoop@<System domain name>. | org.apache.zookeeper.server.auth.SaslServerCallbackHandler.handleAuthorizeCallback(SaslServerCallbackHandler.java:118)
2015-04-15 21:24:54,261 | INFO | NIOServerCxn.Factory:192-168-0-114/192.168.0.114:2181 | Setting authorizedID: hdfs/hadoop@<System domain name> | org.apache.zookeeper.server.auth.SaslServerCallbackHandler.handleAuthorizeCallback(SaslServerCallbackHandler.java:134)
2015-04-15 21:24:54,261 | INFO | NIOServerCxn.Factory:192-168-0-114/192.168.0.114:2181 | adding SASL authorization for authorizationID: hdfs/hadoop@<System domain name> | org.apache.zookeeper.server.ZooKeeperServer.processSasl(ZooKeeperServer.java:1009)
2015-04-15 21:24:54,262 | INFO | ProcessThread(sid:22 cport:-1): | Got user-level KeeperException when processing sessionid:0x164cb2b3e4b36ae4 type:create cxid:0x3 zxid:0x20009fafc txntype:-1 reqpath:n/a Error Path:/hadoop-ha/hacluster/ActiveStandbyElectorLock Error:KeeperErrorCode = NodeExists for /hadoop-ha/hacluster/ActiveStandbyElectorLock | org.apache.zookeeper.server.PrepRequestProcessor.pRequest(PrepRequestProcessor.java:648)
```

Content of the ZKFC log is as follows:

```
2015-04-15 21:24:54,237 | INFO | main-SendThread(192-168-0-114:2181) | Socket connection established to 192-168-0-114/192.168.0.114:2181, initiating session | org.apache.zookeeper.ClientCnxn$SendThread.primeConnection(ClientCnxn.java:854)
```



```
2015-04-15 21:24:54,257 | INFO | main-SendThread(192-168-0-114:2181) | Session establishment complete
on server 192-168-0-114/192.168.0.114:2181, sessionid = 0x144cb2b3e4b36ae4 , negotiated timeout =
45000 | org.apache.zookeeper.ClientCnxn$SendThread.onConnected(ClientCnxn.java:1259)
2015-04-15 21:24:54,260 | INFO | main-EventThread | EventThread shut down |
org.apache.zookeeper.ClientCnxn$EventThread.run(ClientCnxn.java:512)
2015-04-15 21:24:54,262 | INFO | main-EventThread | Session connected. |
org.apache.hadoop.ha.ActiveStandbyElector.processWatchEvent(ActiveStandbyElector.java:547)
2015-04-15 21:24:54,264 | INFO | main-EventThread | Successfully authenticated to ZooKeeper using SASL. |
org.apache.hadoop.ha.ActiveStandbyElector.processWatchEvent(ActiveStandbyElector.java:573)
```

Answer

- Cause Analysis

  After the active NameNode restarts, the temporary node **/hadoop-ha/hacluster/ActiveStandbyElectorLock** created on ZooKeeper is deleted. After the standby NameNode receives that information that the **/hadoop-ha/hacluster/ActiveStandbyElectorLock** node is deleted, the standby NameNode creates the /**hadoop-ha/hacluster/ActiveStandbyElectorLock** node in ZooKeeper in order to switch to the active NameNode. However, when the standby NameNode connects with ZooKeeper through the client ZKFC, the session ID of ZKFC differs from that of ZooKeeper due to network issues, overload CPU, or overload clusters. In this case, the watcher of the standby NameNode fails to detect that the temporary node has been successfully created, and fails to consider the standby NameNode as the active NameNode. After the original active NameNode restarts, it detects that the **/hadoop-ha/hacluster/ActiveStandbyElectorLock** already exists and becomes the standby NameNode. Therefore, both NameNodes are standby NameNodes.

- Solution

  You are advised to restart two ZKFCs of HDFS on FusionInsight Manager.

10.17.7 DataNode Is Normal but Cannot Report Data Blocks

Question

The DataNode is normal, but cannot report data blocks. As a result, the existing data blocks cannot be used.

Answer

This error may occur when the number of data blocks in a data directory exceeds four times the upper limit (4 x 1 MB). And the DataNode generates the following error logs:

```
2015-11-05 10:26:32,936 | ERROR | DataNode:[[[DISK]file:/srv/BigData/hadoop/data1/dn/]] heartbeating to
vm-210/10.91.8.210:8020 | Exception in BPOfferService for Block pool
BP-805114975-10.91.8.210-1446519981645
(Datanode Uuid bcada350-0231-413b-bac0-8c65e906c1bb) service to vm-210/10.91.8.210:8020 |
BPServiceActor.java:824
java.lang.IllegalStateException:com.google.protobuf.InvalidProtocolBufferException:Protocol message was
too large.May
be malicious.Use CodedInputStream.setSizeLimit() to increase the size limit. at
org.apache.hadoop.hdfs.protocol.BlockListAsLongs$BufferDecoder$1.next(BlockListAsLongs.java:369)
at org.apache.hadoop.hdfs.protocol.BlockListAsLongs$BufferDecoder$1.next(BlockListAsLongs.java:347) at
org.apache.hadoop.hdfs.
protocol.BlockListAsLongs$BufferDecoder.getBlockListAsLongs(BlockListAsLongs.java:325) at
org.apache.hadoop.hdfs.protocolPB.DatanodeProtocolClientSideTranslatorPB.
blockReport(DatanodeProtocolClientSideTranslatorPB.java:190) at
```



```
org.apache.hadoop.hdfs.server.datanode.BPServiceActor.blockReport(BPServiceActor.java:473)
at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.offerService(BPServiceActor.java:685) at
org.apache.hadoop.hdfs.server.datanode.BPServiceActor.run(BPServiceActor.java:822)
at java.lang.Thread.run(Thread.java:745) Caused
by:com.google.protobuf.InvalidProtocolBufferException:Protocol message was too large.May be
malicious.Use CodedInputStream.setSizeLimit()
to increase the size limit. at
com.google.protobuf.InvalidProtocolBufferException.sizeLimitExceeded(InvalidProtocolBufferException.java:1
10) at com.google.protobuf.CodedInputStream.refillBuffer(CodedInputStream.java:755)
at com.google.protobuf.CodedInputStream.readRawByte(CodedInputStream.java:769) at
com.google.protobuf.CodedInputStream.readRawVarint64(CodedInputStream.java:462) at
com.google.protobuf.
CodedInputStream.readSInt64(CodedInputStream.java:363) at
org.apache.hadoop.hdfs.protocol.BlockListAsLongs$BufferDecoder$1.next(BlockListAsLongs.java:363)
```

The number of data blocks in the data directory is displayed as **Metric**. You can monitor its value through **http://<datanode-ip>:<http-port>/jmx**. If the value is greater than four times the upper limit (4 x 1 MB), you are advised to configure multiple drives and restart HDFS.

Recovery procedure:

1. Configure multiple data directories on the DataNode.

   For example, configure multiple directories on the DataNode where only the **/data1/datadir** directory is configured:

   ```
   <property> <name>dfs.datanode.data.dir</name> <value>/data1/datadir</value> </property>
   ```

   Configure as follows:

   ```
   <property> <name>dfs.datanode.data.dir</name> <value>/data1/datadir/,/data2/datadir,/data3/
   datadir</value> </property>
   ```

   NOTE

   You are advised to configure multiple data directories on multiple disks. Otherwise, performance may be affected.

2. Restart the HDFS.
3. Perform the following operation to move the data to the new data directory:

   mv */data1/datadir/current/finalized/subdir1 /data2/datadir/current/finalized/subdir1*

4. Restart the HDFS.

10.17.8 Can I Delete or Modify the Data Storage Directory in DataNode?

Question

- In DataNode, the storage directory of data blocks is specified by **dfs.datanode.data.dir.** Can I modify **dfs.datanode.data.dir** to modify the data storage directory?
- Can I modify files under the data storage directory?

Answer

During the system installation, you need to configure the **dfs.datanode.data.dir** parameter to specify one or more root directories.

- During the system installation, you need to configure the dfs.datanode.data.dir parameter to specify one or more root directories.

- Exercise caution when modifying dfs.datanode.data.dir. You can configure this parameter to add a new data root directory.
- Do not modify or delete data blocks in the storage directory. Otherwise, the data blocks will lose.

NOTE

Similarly, do not delete the storage directory, or modify or delete data blocks under the directory using the following parameters:

- dfs.namenode.edits.dir
- dfs.namenode.name.dir
- dfs.journalnode.edits.dir

10.17.9 Failed to Calculate the Capacity of a DataNode when Multiple data.dir Directories Are Configured in a Disk Partition

Question

The capacity of a DataNode fails to calculate when multiple data.dir directories are configured in a disk partition.

Answer

Currently, the capacity is calculated based on disks, which is similar to the ***df*** command in Linux. Ideally, users do not configure multiple data.dir directories in a disk partition. Otherwise, all data will be written to the same disk, greatly deteriorating the performance.

You are advised to configure them as below.

For example, if a node contains the following disks:

```
host-4:~ # df -h
Filesystem      Size  Used  Avail  Use%  Mounted on
/dev/sda1       352G  11G    324G   4%    /
udev            190G  252K  190G   1%    /dev
tmpfs           190G  72K    190G   1%    /dev/shm
/dev/sdb1       2.7T  74G    2.5T   3%    /data1
/dev/sdc1       2.7T  75G    2.5T   3%    /data2
/dev/sdd1       2.7T  73G    2.5T   3%    /da
```

Recommended configuration:

```
<property>
<name>dfs.datanode.data.dir</name>
<value>/data1/datadir/,/data2/datadir,/data3/datadir</value>
</property>
```

Unrecommended configuration:

```
<property>
<name>dfs.datanode.data.dir</name>
<value>/data1/datadir1/,/data2/datadir1,/data3/datadir1,/data1/datadir2,data1/datadir3,/data2/datadir2,/
data2/datadir3,/data3/datadir2,/data3/datadir3</value>
</property>
```


10.17.10 Why Data in the Buffer Is Lost If a Power Outage Occurs During Storage of Small Files

Question

Why data in the buffer is lost if a power outage occurs during storage of small files?

Answer

Because of a power outage, the blocks in the buffer are not written to the disk immediately after the write operation is completed. To enable synchronization of blocks to the disk, set **dfs.datanode.synconclose** to **true** in the *client installation path*/**HDFS/hadoop/etc/hadoop/hdfs-site.xml** file.

By default, **dfs.datanode.synconclose** is set to **false**. This improves the performance but can cause a buffer data loss in the case of a power outage, and therefore, it is recommended that **dfs.datanode.synconclose** be set to **true** even if this may affect the performance. You can determine whether to enable the synchronization function based on your actual situation.

10.17.11 Why Is the Storage Type of File Copies DISK When the Tiered Storage Policy Is LAZY_PERSIST?

Question

When the storage policy of the file is set to **LAZY_PERSIST**, the storage type of the first replica should be **RAM_DISK**, and the storage type of other replicas should be **DISK**.

But why is the storage type of all copies shown as **DISK** actually?

Answer

When a user writes into a file whose storage policy is **LAZY_PERSIST**, three replicas are written one by one. The first replica is preferentially written into the DataNode where the client is located. The storage type of all replicas is **DISK** in the following scenarios:

- If the DataNode where the client is located does not have the RAM disk, the first replica is written into the disk of the DataNode where the client is located, and other replicas are written into the disks of other nodes.
- If the DataNode where the client is located has the RAM disk, and the value of **dfs.datanode.max.locked.memory** is not specified or smaller than the value of **dfs.blocksize** (To obtain the parameter value, log in to FusionInsight Manager and choose **Cluster > Services > HDFS > Configuration > All Configurations**, and search for this parameter.), the first replica is written into the disk of the DataNode where the client is located, and other replicas are written into the disks of other nodes.

10.17.12 Blocks Miss on the NameNode UI After the Successful Rollback

Question

Why are some blocks missing on the NameNode UI after the rollback is successful?

Answer

This problem occurs because blocks with new IDs or genstamps may exist on the DataNode. The block files in the DataNode may have different generation flags and lengths from those in the rollback images of the NameNode. Therefore, the NameNode rejects these blocks in the DataNode and marks the files as damaged.

Scenarios:

1. Before an upgrade:

   Client A writes some data to file X. (Assume A bytes are written.)

2. During an upgrade:

   Client A still writes data to file X. (The data in the file is A + B bytes.)

3. After an upgrade:

   Client A completes the file writing. The final data is A + B bytes.

4. Rollback started:

   The status will be rolled back to the status before the upgrade. That is, file X in NameNode will have A bytes, but block files in DataNode will have A + B bytes.

Recovery procedure:

1. Obtain the list of damaged files from NameNode web UI or run the following command to obtain:

   hdfs fsck <filepath> -list-corruptfileblocks

2. Run the following command to delete unnecessary files:

   hdfs fsck <corrupt file path> - delete

   NOTE

   Deleting a file is a high-risk operation. Ensure that the files are no longer needed before performing this operation.

3. For the required files, run the **fsck** command to obtain the block list and block sequence.
   - In the block sequence table provided, use the block ID to search for the data directory in the DataNode and download the corresponding block from the DataNode.
   - Write all such block files in appending mode based on the sequence to construct the original file.

     Example:

     File 1--> blk_1, blk_2, blk_3

Create a file by combining the contents of all three block files from the same sequence.

- Delete the old file from HDFS and rewrite the new file.

10.18 HDFS Troubleshooting

10.18.1 Why Is "java.net.SocketException: No buffer space available" Reported When Data Is Written to HDFS

Question

Why is an "java.net.SocketException: No buffer space available" exception reported when data is written to HDFS?

This problem occurs when files are written to the HDFS. Check the error logs of the client and DataNode.

The client logs are as follows:

Figure 10-14 Client logs

```
2017-07-05 21:58:06,459 INFO  [htable-pool3-t1] ipc.AbstractRpcClient: RPC Server Kerberos principal name for service=ClientService is hbase/hadoop.hadoop123.com@HADOOP12
2017-07-05 21:58:06,893 WARN  [main] mapreduce.LoadIncrementalHFiles: Skipping non-directory hdfs://hacluster/HBaseTest/bulkload_output/_SUCCESS
2017-07-05 21:59:13,211 WARN  [main] hdfs.BlockReaderFactory: I/O error constructing remote block reader.
java.net.SocketException: No buffer space available
        at sun.nio.ch.Net.connect0(Native Method)
        at sun.nio.ch.Net.connect(Net.java:454)
        at sun.nio.ch.Net.connect(Net.java:446)
        at sun.nio.ch.SocketChannelImpl.connect(SocketChannelImpl.java:648)
        at org.apache.hadoop.net.SocketIOWithTimeout.connect(SocketIOWithTimeout.java:192)
        at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:531)
        at org.apache.hadoop.hdfs.DFSClient.newConnectedPeer(DFSClient.java:3345)
        at org.apache.hadoop.hdfs.BlockReaderFactory.nextTcpPeer(BlockReaderFactory.java:789)
        at org.apache.hadoop.hdfs.BlockReaderFactory.getRemoteBlockReaderFromTcp(BlockReaderFactory.java:706)
        at org.apache.hadoop.hdfs.BlockReaderFactory.build(BlockReaderFactory.java:369)
        at org.apache.hadoop.hdfs.DFSInputStream.getBlockReader(DFSInputStream.java:713)
        at org.apache.hadoop.hdfs.DFSInputStream.blockSeekTo(DFSInputStream.java:663)
        at org.apache.hadoop.hdfs.DFSInputStream.readWithStrategy(DFSInputStream.java:919)
        at org.apache.hadoop.hdfs.DFSInputStream.read(DFSInputStream.java:973)
        at java.io.DataInputStream.readFully(DataInputStream.java:195)
        at org.apache.hadoop.hbase.io.hfile.FixedFileTrailer.readFromStream(FixedFileTrailer.java:391)
        at org.apache.hadoop.hbase.io.hfile.HFile.isHFileFormat(HFile.java:578)
        at org.apache.hadoop.hbase.io.hfile.HFile.isHFileFormat(HFile.java:560)
        at org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles.visitBulkHFiles(LoadIncrementalHFiles.java:229)
        at org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles.discoverLoadQueue(LoadIncrementalHFiles.java:281)
        at org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles.prepareHFileQueue(LoadIncrementalHFiles.java:452)
        at org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles.doBulkLoad(LoadIncrementalHFiles.java:365)
        at org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles.doBulkLoad(LoadIncrementalHFiles.java:331)
        at org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles.run(LoadIncrementalHFiles.java:1107)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70)
        at org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles.main(LoadIncrementalHFiles.java:1114)
2017-07-05 21:59:13,215 WARN  [main] hdfs.DFSClient: Failed to connect to /192.168.152.128:25009 for block BP-1989348819-192.168.199.5-1497961637591:blk_1107301222_335745
ffer space available
java.net.SocketException: No buffer space available
        at sun.nio.ch.Net.connect0(Native Method)
        at sun.nio.ch.Net.connect(Net.java:454)
        at sun.nio.ch.Net.connect(Net.java:446)
        at sun.nio.ch.SocketChannelImpl.connect(SocketChannelImpl.java:648)
        at org.apache.hadoop.net.SocketIOWithTimeout.connect(SocketIOWithTimeout.java:192)
        at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:531)
        at org.apache.hadoop.hdfs.DFSClient.newConnectedPeer(DFSClient.java:3345)
```

DataNode logs are as follows:

```
2017-07-24 20:43:39,269 | ERROR | DataXceiver for client DFSClient_NONMAPREDUCE_996005058_86
 at /192.168.164.155:40214 [Receiving block
BP-1287143557-192.168.199.6-1500707719940:blk_1074269754_528941 with io weight 10] |
DataNode{data=FSDataset{dirpath='[/srv/BigData/hadoop/data1/dn/current, /srv/BigData/hadoop/
data2/dn/current, /srv/BigData/hadoop/data3/dn/current, /srv/BigData/hadoop/data4/dn/current, /srv/
BigData/hadoop/data5/dn/current, /srv/BigData/hadoop/data6/dn/current, /srv/BigData/hadoop/data7/dn/
current]'}, localName='192-168-164-155:9866', datanodeUuid='a013e29c-4e72-400c-bc7b-bbbf0799604c',
xmitsInProgress=0}:Exception transfering block
BP-1287143557-192.168.199.6-1500707719940:blk_1074269754_528941 to mirror 192.168.202.99:9866:
java.net.SocketException: No buffer space available | DataXceiver.java:870
2017-07-24 20:43:39,269 | INFO | DataXceiver for client DFSClient_NONMAPREDUCE_996005058_86
 at /192.168.164.155:40214 [Receiving block
BP-1287143557-192.168.199.6-1500707719940:blk_1074269754_528941 with io weight 10] | opWriteBlock
BP-1287143557-192.168.199.6-1500707719940:blk_1074269754_528941 received exception
java.net.SocketException: No buffer space available | DataXceiver.java:933
2017-07-24 20:43:39,270 | ERROR | DataXceiver for client DFSClient_NONMAPREDUCE_996005058_86
```



```
 at /192.168.164.155:40214 [Receiving block
BP-1287143557-192.168.199.6-1500707719940:blk_1074269754_528941 with io weight 10] |
192-168-164-155:9866:DataXceiver error processing WRITE_BLOCK operation src: /192.168.164.155:40214
dst: /192.168.164.155:9866 | DataXceiver.java:304 java.net.SocketException: No buffer space available
 at sun.nio.ch.Net.connect0(Native Method)
 at sun.nio.ch.Net.connect(Net.java:454)
 at sun.nio.ch.Net.connect(Net.java:446)
 at sun.nio.ch.SocketChannelImpl.connect(SocketChannelImpl.java:648)
 at org.apache.hadoop.net.SocketIOWithTimeout.connect(SocketIOWithTimeout.java:192)
 at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:531)
 at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:495)
 at org.apache.hadoop.hdfs.server.datanode.DataXceiver.writeBlock(DataXceiver.java:800)
 at org.apache.hadoop.hdfs.protocol.datatransfer.Receiver.opWriteBlock(Receiver.java:138)
 at org.apache.hadoop.hdfs.protocol.datatransfer.Receiver.processOp(Receiver.java:74)
 at org.apache.hadoop.hdfs.server.datanode.DataXceiver.run(DataXceiver.java:265)
 at java.lang.Thread.run(Thread.java:748)
```

Answer

The preceding problem may be caused by network memory exhaustion.

You can increase the threshold of the network device based on the actual scenario.

Example:

```
[root@xxxxx ~]# cat /proc/sys/net/ipv4/neigh/default/gc_thresh*
128
512
1024
[root@xxxxx ~]# echo 512 > /proc/sys/net/ipv4/neigh/default/gc_thresh1
[root@xxxxx ~]# echo 2048 > /proc/sys/net/ipv4/neigh/default/gc_thresh2
[root@xxxxx ~]# echo 4096 > /proc/sys/net/ipv4/neigh/default/gc_thresh3
[root@xxxxx ~]# cat /proc/sys/net/ipv4/neigh/default/gc_thresh*
512
2048
4096
```

You can also add the following parameters to the **/etc/sysctl.conf** file. The configuration takes effect even if the host is restarted.

```
net.ipv4.neigh.default.gc_thresh1 = 512
net.ipv4.neigh.default.gc_thresh2 = 2048
net.ipv4.neigh.default.gc_thresh3 = 4096
```

10.18.2 NameNode Startup Is Slow

Question

The NameNode startup is slow when it is restarted immediately after a large number of files (for example, 1 million files) are deleted.

Answer

It takes time for the DataNode to delete the corresponding blocks after files are deleted. When the NameNode is restarted immediately, it checks the block information reported by all DataNodes. If a deleted block is found, the NameNode generates the corresponding INFO log information, as shown below:

```
2015-06-10 19:25:50,215 | INFO  | IPC Server handler 36 on 25000 | BLOCK* processReport:
blk_1075861877_2121067 on node 10.91.8.218:9866 size 10249 does not belong to any file |
org.apache.hadoop.hdfs.server.blockmanagement.BlockManager.processReport(BlockManager.java:1854)
```

A log is generated for each deleted block. A file may contain one or more blocks. Therefore, after startup, the NameNode spends a large amount of time printing

logs when a large number of files are deleted. As a result, the NameNode startup becomes slow.

To address this issue, the following operations can be performed to speed up the startup:

1. After a large number of files are deleted, wait until the DataNode deletes the corresponding blocks and then restart the NameNode.

   You can run the ***hdfs dfsadmin -report*** command to check the disk space and check whether the files have been deleted.

2. If a large number of the preceding logs are generated, you can change the NameNode log level to **ERROR** so that the NameNode stops printing such logs.

   After the NameNode is restarted, change the log level back to **INFO**. You do not need to restart the service after changing the log level.

10.18.3 NameNode Fails to Be Restarted Due to EditLog Discontinuity

Question

If a JournalNode server is powered off, the data directory disk is fully occupied, and the network is abnormal, the EditLog sequence number on the JournalNode is inconsecutive. In this case, the NameNode restart may fail.

Symptom

The NameNode fails to be restarted. The following error information is reported in the NameNode run logs:

```
2019-11-08 16:30:28,399 | ERROR | main | Failed to start namenode. | NameNode.java:1732
java.io.IOException: There appears to be a gap in the edit log.  We expected txid 13698019, but got txid 13698088.
        at org.apache.hadoop.hdfs.server.namenode.MetaRecoveryContext.editLogLoaderPrompt(MetaRecoveryContext.java:94)
        at org.apache.hadoop.hdfs.server.namenode.FSEditLogLoader.loadEditRecords(FSEditLogLoader.java:278)
        at org.apache.hadoop.hdfs.server.namenode.FSEditLogLoader.loadFSEdits(FSEditLogLoader.java:188)
        at org.apache.hadoop.hdfs.server.namenode.FSImage.loadEdits(FSImage.java:924)
        at org.apache.hadoop.hdfs.server.namenode.FSImage.loadFSImage(FSImage.java:771)
        at org.apache.hadoop.hdfs.server.namenode.FSImage.recoverTransitionRead(FSImage.java:331)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFSImage(FSNamesystem.java:1108)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFromDisk(FSNamesystem.java:727)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.loadNamesystem(NameNode.java:638)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.initialize(NameNode.java:700)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:943)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:916)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.createNameNode(NameNode.java:1655)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.main(NameNode.java:1725)
```

Solution

1. Find the active NameNode before the restart, go to its data directory (you can obtain the directory, such as **/srv/BigData/namenode/current** by checking the configuration item **dfs.namenode.name.dir**), and obtain the sequence number of the latest FsImage file, as shown in the following figure:


```
-rw-------. 1 omm wheel     574 Oct  2 01:12 edits_0000000000013259401-0000000000
-rw-------. 1 omm wheel     575 Oct  2 01:13 edits_0000000000013259409-0000000000
-rw-------. 1 omm wheel      42 Oct  2 01:13 edits_0000000000013259417-0000000000
-rw-------. 1 omm wheel 1048576 Nov  8 16:01 edits_inprogress_0000000000013698088
-rw-------. 1 omm wheel  314803 Nov  8 15:53 fsimage_0000000000013698018
-rw-------. 1 omm wheel      62 Nov  8 15:53 fsimage_0000000000013698018.md5
-rw-------. 1 omm wheel  314803 Nov  8 15:56 fsimage_0000000000013698050
-rw-------. 1 omm wheel      62 Nov  8 15:56 fsimage_0000000000013698050.md5
-rw-------. 1 omm wheel  314803 Nov  8 15:59 fsimage_0000000000013698066
-rw-------. 1 omm wheel      62 Nov  8 15:59 fsimage_0000000000013698066.md5
-rw-------. 1 omm wheel       9 Oct  2 01:13 seen_txid
-rw-------. 1 omm wheel     187 Nov  8 15:59 VERSION
```

2. Check the data directory of each JournalNode (you can obtain the directory such as**/srv/BigData/journalnode/hacluster/current** by checking the value of the configuration item **dfs.journalnode.edits.dir**), and check whether the sequence number starting from that obtained in step 1 is consecutive in edits files. That is, you need to check whether the last sequence number of the previous edits file is consecutive with the first sequence number of the next edits file. (As shown in the following figure, edits_0000000000013259231-0000000000013259237 and edits_0000000000013259239-0000000000013259246 are not consecutive.)

```
-rw-------. 1 omm wheel     575 Oct  2 00:43 edits_0000000000013259159-0000000000013259166
-rw-------. 1 omm wheel     576 Oct  2 00:43 edits_0000000000013259167-0000000000013259174
-rw-------. 1 omm wheel     575 Oct  2 00:45 edits_0000000000013259175-0000000000013259182
-rw-------. 1 omm wheel     575 Oct  2 00:45 edits_0000000000013259183-0000000000013259190
-rw-------. 1 omm wheel     576 Oct  2 00:47 edits_0000000000013259191-0000000000013259198
-rw-------. 1 omm wheel     575 Oct  2 00:48 edits_0000000000013259199-0000000000013259206
-rw-------. 1 omm wheel     575 Oct  2 00:49 edits_0000000000013259207-0000000000013259214
-rw-------. 1 omm wheel     575 Oct  2 00:50 edits_0000000000013259215-0000000000013259222
-rw-------. 1 omm wheel     573 Oct  2 00:51 edits_0000000000013259223-0000000000013259230
-rw-------. 1 omm wheel     571 Oct  2 00:52 edits_0000000000013259231-0000000000013259237
-rw-------. 1 omm wheel     576 Oct  2 00:53 edits_0000000000013259239-0000000000013259246
-rw-------. 1 omm wheel     575 Oct  2 00:54 edits_0000000000013259247-0000000000013259254
-rw-------. 1 omm wheel     576 Oct  2 00:55 edits_0000000000013259255-0000000000013259262
-rw-------. 1 omm wheel      42 Oct  2 00:56 edits_0000000000013259263-0000000000013259264
-rw-------. 1 omm wheel    1107 Oct  2 00:57 edits_0000000000013259265-0000000000013259278
-rw-------. 1 omm wheel      42 Oct  2 00:58 edits_0000000000013259279-0000000000013259280
-rw-------. 1 omm wheel    1109 Oct  2 00:59 edits_0000000000013259281-0000000000013259294
-rw-------. 1 omm wheel      42 Oct  2 01:00 edits_0000000000013259295-0000000000013259296
-rw-------. 1 omm wheel    1299 Oct  2 01:01 edits_0000000000013259297-0000000000013259312
-rw-------. 1 omm wheel     260 Oct  2 01:02 edits_0000000000013259313-0000000000013259316
-rw-------. 1 omm wheel     984 Oct  2 01:03 edits_0000000000013259317-0000000000013259328
-rw-------. 1 omm wheel     572 Oct  2 01:04 edits_0000000000013259329-0000000000013259336
-rw-------. 1 omm wheel     575 Oct  2 01:05 edits_0000000000013259337-0000000000013259344
```

3. If the edits files are not consecutive, check whether the edits files with the related sequence number exist in the data directories of other JournalNodes or NameNode. If the edits files can be found, copy a consecutive segment to the JournalNode.
4. In this way, all inconsecutive edits files are restored.
5. Restart the NameNode and check whether the restart is successful. If the fault persists, contact technical support.

10.18.4 Standby NameNode Fails to Be Restarted When the System Is Powered off During Metadata (Namespace) Storage

Question

When the standby NameNode is powered off during metadata (namespace) storage, it fails to be started and the following error information is displayed.


```
2015-12-04 11:49:12,121 | ERROR | main | Failed to load image from FS
ImageFile(file=/srv/BigData/namenode/current/fsimage_0000000000000096
080,
cpktTxId=0000000000000096080) | FSImage.java:685
java.io.IOException: Invalid MD5 file /srv/BigData/namenode/current/f
simage_0000000000000096080.md5:
the content "                                  does not match the expecte
d pattern.
at org.apache.hadoop.hdfs.util.MD5FileUtils.readStoredMd5(MD5FileUtil
s.java:92)
at org.apache.hadoop.hdfs.util.MD5FileUtils.readStoredMd5ForFile(MD5F
ileUtils.java:109)
at org.apache.hadoop.hdfs.server.namenode.FSImage.loadFSImage(FSImage
.java:975)
at org.apache.hadoop.hdfs.server.namenode.FSImage.loadFSImageFile(FSI
mage.java:744)
at org.apache.hadoop.hdfs.server.namenode.FSImage.loadFSImage(FSImage
.java:682)
at org.apache.hadoop.hdfs.server.namenode.FSImage.recoverTransitionRe
ad(FSImage.java:300)
at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFSImage(FS
Namesystem.java:968)
at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFromDisk(F
SNamesystem.java:675)
at org.apache.hadoop.hdfs.server.namenode.NameNode.loadNamesystem(Nam
eNode.java:625)
at org.apache.hadoop.hdfs.server.namenode.NameNode.initialize(NameNod
e.java:685)
at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.ja
va:889)
at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.ja
va:872)
at org.apache.hadoop.hdfs.server.namenode.NameNode.createNameNode(Nam
eNode.java:1580)
at org.apache.hadoop.hdfs.server.namenode.NameNode.main(NameNode.java
:1654)
```

Answer

When the standby NameNode is powered off during metadata (namespace) storage, it fails to be started and the MD5 file is damaged. Remove the damaged fsimage and start the standby NameNode to rectify the fault. After the rectification, the standby NameNode loads the previous fsimage and reproduces all edits.

Recovery procedure:

1. Run the following command to remove the damaged fsimage:

   rm -rf ${BIGDATA_DATA_HOME}/namenode/current/fsimage_0000000000000096

2. Start the standby NameNode.

10.18.5 Why Does DataNode Fail to Start When the Number of Disks Specified by dfs.datanode.data.dir Equals dfs.datanode.failed.volumes.tolerated?

Question

If the number of disks specified by **dfs.datanode.data.dir** is equal to the value of **dfs.datanode.failed.volumes.tolerated**, DataNode startup will fail.

Answer

By default, the failure of a single disk will cause the HDFS DataNode process to shut down, which results in the NameNode scheduling additional replicas for each block that is present on the DataNode. This causes needless replications of blocks that reside on disks that have not failed.

To prevent this, you can configure DataNodes to tolerate the failure of dfs.data.dir directories; Log in to FusionInsight Manager, choose **Cluster > Services > HDFS > Configuration > All Configurations**, and search for **dfs.datanode.failed.volumes.tolerated.** For example, if the value for this parameter is 3, the DataNode will only shut down after four or more data directories have failed. This value is respected on DataNode startup.

When we are configuring tolerate volumes which should be always less than the configured volumes or else we can keep this as -1 which is equal to n-1 (where n is number of disks) then DataNode will not be shut down.

10.18.6 Why Does Array Border-crossing Occur During FileInputFormat Split?

Question

When HDFS calls the FileInputFormat getSplit method, the ArrayIndexOutOfBoundsException: 0 appears in the following log:

```
java.lang.ArrayIndexOutOfBoundsException: 0
at org.apache.hadoop.mapred.FileInputFormat.identifyHosts(FileInputFormat.java:708)
at org.apache.hadoop.mapred.FileInputFormat.getSplitHostsAndCachedHosts(FileInputFormat.java:675)
at org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:359)
at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:210)
at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:239)
at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:237)
at scala.Option.getOrElse(Option.scala:120)
at org.apache.spark.rdd.RDD.partitions(RDD.scala:237)
at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:35)
```

Answer

The elements of each block correspondent frame are as below: /default/rack0/:,/default/rack0/datanodeip:port.

The problem is due to a block damage or loss, making the block correspondent machine ip and port become null. Use **hdfs fsck** to check the file blocks health state when this problem occurs, and remove damaged block or restore the missing block to re-computing the task.

10.18.7 The Standby NameNode Fails to Be Started Because It Is Not Started for a Long Time

Symptom

The standby NameNode is not started for a long time. After the **edits** file is automatically deleted due to the aging policy, this file cannot be found when the NameNode is restarted. As a result, an error is reported.


```
There appears to be a gap in the edit log. We expected txid XXX, but got txid XXX.
```

Solution

Step 1 Go to the **All Configurations** page of HDFS by referring to **Modifying Cluster Service Configuration Parameters**, search for **dfs.namenode.name.dir** and check the value to obtain the NameNode data directory, for example, **/srv/BigData/namenode/current**.

Step 2 On the HDFS service page, click the **Instances** tab to view and record the service IP addresses of the active and standby NameNodes.

Step 3 Log in to the faulty standby NameNode as user **root** and back up the **fsimage** file in the data directory obtained in **Step 1**. For example, back up the data to the **/srv/BigData/namenode/current.bak** directory.

mv /srv/BigData/namenode/current/ /srv/BigData/namenode/current.bak

Step 4 Log in to the active NameNode as the **root** user and run the following command to copy the **fsimage** file to the standby NameNode:

scp -rp /srv/BigData/namenode/current/ {*IP address of the standby NameNode***}:/srv/BigData/namenode/**

chown omm:wheel /srv/BigData/namenode/current -R

Step 5 Restart the standby NameNode and check whether restart is successful. If the operation fails, contact technical support.

----End

11 Using HetuEngine

11.1 Overview of HetuEngine Interactive Query

HetuEngine supports quick joint query of multiple data sources and GUI-based data source configuration and management. You can quickly add a data source on the HSConsole page.

The following table lists the data sources supported by HetuEngine of the current version.

Table 11-1 List for connecting HetuEngine to data sources

HetuEngine Mode	Data Source	Data Source Mode	Supported Data Source Version
Security mode	Hive	Security mode	MRS 3.x and FusionInsight 6.5.1
	HBase		MRS 3.x and FusionInsight 6.5.1
	HetuEngine		MRS 3.1.1 and later
	GaussDB		GaussDB 200 and GaussDB A 8.0.0 and later
	Hudi		MRS 3.1.2 and later
	ClickHouse		MRS 3.1.1 and later
	IoTDB		MRS 3.2.0 and later
	MySQL		MySQL 5.7, MySQL 8.0, and later
	Oracle		Oracle 12 and later
	GBase		GBase8a V950 and later
Normal mode	Hive	Normal mode	MRS 3.x and FusionInsight 6.5.1
	HBase		MRS 3.x and FusionInsight 6.5.1

HetuEngine Mode	Data Source	Data Source Mode	Supported Data Source Version
	Hudi		MRS 3.1.2 and later
	ClickHouse		MRS 3.1.1 and later
	IoTDB		MRS 3.2.0 and later
	GaussDB	Security Mode	GaussDB 200 and GaussDB A 8.0.0 and later
	MySQL		MySQL 5.7, MySQL 8.0 and later
	Oracle		Oracle 12 and later
	GBase		GBase8a V950 and later

Operations such as adding, configuring, and deleting a HetuEngine data source takes effect dynamically without restarting the cluster.

A configured data source takes effect dynamically and you cannot disable this function. By default, the interval for a data source to dynamically take effect is 60 seconds. You can change the interval to a desired one by changing the value of **catalog.scanner-interval** in **coordinator.config.properties** and **worker.config.properties** by referring to **Step 3.5** in **Creating a HetuEngine Compute Instance**. See the following example.

```
catalog.scanner-interval =120s
```

HetuEngine supports query pushdown. It can push down queries or partial queries to connected data sources. This means that special predicates, aggregate functions, or other operations can be passed to the underlying database or file system for processing. Query pushdown brings the following benefits:

1. Improves the overall query performance.
2. Reduces the network traffic between HetuEngine and data sources.
3. Reduces the load of remote data sources.

Whether HetuEngine supports query pushdown depends on specific connectors and the underlying data sources or storage systems related to the connectors.

NOTE

- The data source cluster and the HetuEngine cluster must use different domain names. Two data sources (Hive, HBase, and Hudi) with the same domain name cannot be connected to HetuEngine at the same time.
- Nodes in the data source cluster and the HetuEngine cluster can communicate with each other on the service plane.

11.2 HetuEngine User Permission Management

11.2.1 HetuEngine User Permissions

HetuEngine provides the following two permission control models when Kerberos authentication is enabled for the cluster (the cluster is in security mode). By default, the Ranger permission model is used. When Kerberos authentication is disabled for the cluster (the cluster is in normal mode), the Ranger permission model is provided but disabled by default.

- For details about the Ranger model, see **HetuEngine Ranger-based Permission Control**.
- For details about the MetaStore model, see **HetuEngine MetaStore-based Permission Control**.

The following table lists the differences between Ranger and MetaStore. Both Ranger and MetaStore support user, user group, and role authentication.

Table 11-2 Differences between Ranger and MetaStore

Permission Control Mode	Permission Model	Supported Data Source	Description
Ranger	PBAC	Hive, HBase, Elasticsearch, GaussDB, HetuEngine, ClickHouse, IoTDB, Hudi, MySQL	Row filtering, column masking, and fine-grained permission control are supported.
MetaStore	RBAC	Hive	-

Permission Principles and Constraints

- Accessing data sources in the same cluster using HetuEngine

  If Ranger authentication is enabled for HetuEngine, the PBAC permission policy of Ranger is used for authentication.

  If Ranger authentication is disabled for HetuEngine, the RBAC permission policy of MetaStore is used for authentication.
- Accessing data sources in different clusters using HetuEngine

  The permission policy is controlled by the permissions of the HetuEngine client and the data source. (In Hive scenarios, it depends on HDFS.)
- When querying a view, you only need to grant the select permission on the target view. When querying a join table using a view, you need to grant the select permission on the view and table.
- Columns in GaussDB and HetuEngine data sources cannot be masked.

NOTE

When the permission control type of HetuEngine is changed, the HetuEngine service, including the HetuEngine compute instance running on the HSConsole page, needs to be restarted.

HetuEngine Ranger-based Permission Control

By default, Ranger authentication is used for newly installed clusters. For clusters upgraded from earlier versions or clusters where Ranger authentication is manually disabled, you can enable Ranger authentication again by referring to the following content.

For a cluster with Ranger authentication enabled, cluster administrators can use Ranger to configure the permissions to manage databases, tables, and columns of data sources for HetuEngine users. For details, see **Adding a Ranger Access Permission Policy for HetuEngine**.

Step 1 Log in to FusionInsight Manager.

Step 2 If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), add the **ranger.usersync.sync.source** parameter. If Kerberos authentication is enabled for the cluster (the cluster is in security mode), skip this step.

1. Choose **Cluster** > **Services** > **Ranger**. Click **Configurations** then **All Configurations**.
2. Search for the **ranger.usersync.config.expandor** parameter, set its name to **ranger.usersync.sync.source**, set its value to **ldap**, and save the settings.
3. On the **Dashboard** page, click **More** > **Restart Service** in the upper right corner, enter the password, and restart Ranger.

   NOTE

   For MRS 3.5.0 and later versions, run the following commands:

   To use Ranger for permission control when Kerberos authentication is disabled for the cluster (the cluster is in normal mode), choose Cluster > **Services** > **Ranger** > **Configurations** > **All Configurations**, search for **ranger.usersync.sync.source**, and ensure that the value is **ldap**, otherwise, change the value to **ldap**. Save the modification and restart Ranger.

Step 3 Choose **Cluster** > **Services** > **HetuEngine** > **More** > **Enable Ranger**.

Step 4 Choose **Cluster** > **Services** > **HetuEngine**. Click **More** and select **Restart Service**.

Step 5 Restart the compute instance on HSConsole.

----End

HetuEngine MetaStore-based Permission Control

- Constraints: This function applies only to Hive data sources.

  When multiple HetuEngine clusters are deployed for collaborative computing, the metadata is centrally managed by the management cluster. Data computing is performed in all clusters. The user permission for accessing HetuEngine clusters must be configured in the management cluster. Users who belong to the Hive user group and share the same name are added to all compute instances.

- Enabling MetaStore Authentication

  a. Log in to FusionInsight Manager.

  b. Choose **Cluster** > **Services** > **HetuEngine**. Click **More** and select **Disable Ranger**.

c. Choose **Cluster** > **Services** > **HetuEngine**. Click **More** and select **Restart Service**.

d. Restart the compute instance on the HSConsole page.

- MetaStore Permission

  Similar to Hive, HetuEngine is a data warehouse framework built on Hadoop, providing storage of structured data like SQL.

  Permissions in a cluster must be assigned to roles which are associated to users or user groups. Users can obtain permissions only by binding a role or joining a group that is bound with a role.

  HetuEngine permission management is performed by the permission system to manage users' operations on the database, ensuring that different users can operate databases independently and securely. A user can operate another user's tables and databases only with the corresponding permissions. Otherwise, operations will be rejected.

  HetuEngine permission management integrates the functions of Hive permission management. MetaStore service of Hive and the function of granting permissions on the web page are required to enable the HetuEngine permission management.

  - Granting permissions on the web page: HetuEngine supports only granting permissions on the web page. On Manager, choose **System** > **Permission** to add or delete a user, user group, or a role, and to grant or cancel permissions.
  - Obtaining and judging a service: When the DDL and DML commands are received from the client, HetuEngine will obtain the client user's permissions on database information from MetaStore, and check whether the required permissions are included. If the required permissions have been obtained, the user's operations are allowed. If the permissions are not obtained, the user's operation will be rejected. After the MetaStore permission check is passed, ACL permission also needs to be checked on HDFS.

- HetuEngine Permission Model

  If a user uses HetuEngine to perform SQL query, the user must be granted with permissions of HetuEngine databases and tables (include external tables and views). The complete permission model of HetuEngine consists of the metadata permission and HDFS file permission. Permissions required to use a database or a table are just one type of HetuEngine permission.

  - Metadata permissions

    Metadata permissions are controlled at the metadata level. Similar to traditional relational databases, the HetuEngine database contains the CREATE and SELECT permissions. Tables and columns contain the SELECT, INSERT, UPDATE, and DELETE permissions. HetuEngine also supports the owner permission OWNERSHIP and cluster administrator permission ADMIN.

  - Data file permissions (that is, HDFS file permissions)

    HetuEngine database and table files are stored in HDFS. The created databases or tables are saved in the **/user/hive/warehouse** directory of HDFS by default. The system automatically creates subdirectories named after database names and database table names. To access a database or

a table, the corresponding file permissions (read, write, and execute) on the HDFS are required.

To perform various operations on HetuEngine databases or tables, you need to associate the metadata permission and the HDFS file permission. For example, to query HetuEngine data tables, you need to associate the metadata permission SELECT with the READ and EXECUTE permissions on HDFS files.

To use the management function of FusionInsight Manager GUI to manage the permissions of HetuEngine databases and tables, you only need to configure the metadata permission, and the system will automatically associate and configure the HDFS file permission. In this way, operations on the interface are simplified, improving efficiency.

- HetuEngine Application Scenarios and Related Permissions

  A user needs to join in the Hive group if a database is created using the HetuEngine service, and role authorization is not required. Users have all permissions on the databases or tables created by themselves in Hive or HDFS. They can create tables, select, delete, insert, or update data, and grant permissions to other users to allow them to access the tables and corresponding HDFS directories and files.

  A user can access the tables or database only with permissions. Permissions required for the user vary depending on different HetuEngine scenarios.

  Table 11-3 Typical HetuEngine scenarios and required permissions

Typical Scenario	Required Permission
Using HetuEngine tables, columns, or databases	Permissions required in different scenarios are as follows: • To create a table, the CREATE permission is required. • To query data, the SELECT permission is required. • To insert data, the INSERT permission is required.

  In some special HetuEngine scenarios, other permissions must be configured separately.

Table 11-4 Typical HetuEngine authentication scenarios and required permissions

Scenario	Required Permission
Creating HetuEngine databases, tables, and foreign tables, or adding partitions to created tables or foreign tables when data files specified by Hive users are saved to other HDFS directories except **/user/hive/warehouse**.	The directory must exist, the client user must be the owner of the directory, and the user must have the **Read**, **Write**, and **Execute** permissions on the directory. The user must have the **Read** and **Execute** permissions of all the upper-layer directories of the directory.
Performing operations on all databases and tables in Hive	The user must be added to the **supergroup** user group, and be assigned the ADMIN permission.

- Configuring Permissions for SparkSQL Tables, Columns, and Databases

  After MetaStore authentication is enabled, if a user needs to access HetuEngine tables or databases created by other users, the user needs to be granted with related permissions. HetuEngine supports permission control based on columns for strict permission control. If a user needs to access some columns in tables created by other users, the user must be granted the permission for columns.

  NOTE

  - Any permission for a table in the database is automatically associated with the HDFS permission for the database directory to facilitate permission management. When any permission for a table is canceled, the system does not automatically cancel the HDFS permission for the database directory to ensure performance. In this case, users can only log in to the database and view table names.
  - When the query permission on a database is added to or deleted from a role, the query permission on tables in the database is automatically added to or deleted from the role. This mechanism is inherited from Hive.
  - In HetuEngine, the name of a column of the **struct** type data cannot contain special characters, that is, characters other than letters, digits, and underscores (_). If the column name of the struct data type contains special characters, the column cannot be displayed on the FusionInsight Manager console when you grant permissions to roles on the **Role** page.

  Procedure

  a. Log in to FusionInsight Manager.
  b. Choose **System** > **Permission** > **Role**.
  c. Click **Create Role**, and set **Role Name** and **Description**.
  d. In the **Configure Resource Permission** area, choose *Name of the desired cluster* > **Hive** and set role permissions. For details, see **Table 11-5**.

     - **Hive Admin Privilege**: Hive administrator permission.
     - **Hive Read Write Privileges**: Hive data table management permission, which is the operation permission to set and manage the data of created tables.

NOTE

- Hive role management supports the administrator permission, and the permissions of accessing tables and views, without granting the database permission.
- The permissions of the Hive administrator do not include the permission to manage HDFS.
- If there are too many tables in the database or too many files in tables, the permission granting may last a while. For example, if a table contains 10,000 files, the permission granting lasts about 2 minutes.

Table 11-5 Setting a role

Task	Operation
Setting the permission to query a table of another user in the default database	1. In the **View Name** area, click **Hive Read Write Privileges**. 2. Click the name of the specified database in the database list. Tables in the database are displayed. 3. In the **Permission** column of a specified table, choose **Select**.
Setting the permission to import data to a table of another user in the default database	1. In the **View Name** area, click **Hive Read Write Privileges**. 2. Click the name of the specified database in the database list. Tables in the database are displayed. 3. In the **Permission** column of the specified indexes, select **Delete** and **Insert**.

e. Click **OK**. Return to the **Role** page.

NOTE

After the role is created, you can create a HetuEngine user and assign related role permissions to the user by referring to **Creating a HetuEngine Permission Role**.

Table 11-6 describes the permission requirements when SQL statements are processed in HetuEngine.

Table 11-6 Using HetuEngine tables, columns, or data

Scenario	Required Permission
DESCRIBE TABLE	Select
ANALYZE TABLE	Select and Insert
SHOW COLUMNS	Select
SHOW TABLE STATUS	Select

Scenario	Required Permission
SHOW TABLE PROPERTIES	Select
SELECT	Select
EXPLAIN	Select
CREATE VIEW	Select, Grant Of Select, and Create
CREATE TABLE	Create
ALTER TABLE ADD PARTITION	Insert
INSERT	Insert
INSERT OVERWRITE	Insert and Delete
ALTER TABLE DROP PARTITION	Table-level Alter and Delete, and column-level Select
ALTER DATABASE	Hive Admin Privilege

11.2.2 Creating a HetuEngine Permission Role

Before using the HetuEngine service in a security cluster, a cluster administrator needs to create a user and grant operation permissions to the user to meet service requirements.

HetuEngine users are classified into administrators and common users. The default HetuEngine administrator group is **hetuadmin**, and the user group of HetuEngine common users is **hetuuser**.

- Users associated with the **hetuadmin** user group can obtain the O&M administrator permissions on the HetuEngine HSConsole web UI and HetuEngine compute instance web UI.
- Users associated with the **hetuuser** user group can obtain the SQL execution permission. They also have permissions to access the HSConsole web UI, view information about clusters of associated tenants and basic information about all data sources, access the web UI of compute instances, and query and maintain SQL statements of the current user.

If Ranger authentication is enabled and you need to configure the permissions to manage databases, tables, and columns of data sources for a user after it is created, see **Adding a Ranger Access Permission Policy for HetuEngine**.

Before you use the HetuEngine service, ensure that the tenant to be associated with the HetuEngine user has been planned and created.

Creating a HetuEngine User

Creating a HetuEngine administrator

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **System** > **Permission** > **User** > **Create**.

Step 3 Enter a username, for example, **hetu_admin**.

Step 4 Set **User Type** to **Human-machine**.

Step 5 Set **Password** and confirm your password.

Step 6 In the **User Group** area, click **Add** to add the **hive**, **hetuadmin**, **hadoop**, **hetuuser**, and **yarnviewgroup** user groups for the user.

Step 7 In the **Primary Group** drop-down list, select **hive** as the primary group.

Step 8 In the **Role** area, click **Add** to assign the **default**, **System_administrator**, and desired tenant role permissions to the user.

Step 9 Click **OK**.

----End

Creating a common HetuEngine user

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **System** > **Permission** > **User** > **Create**.

Step 3 Enter a username, for example, **hetu_test**.

Step 4 Set **User Type** to **Human-machine**.

Step 5 Set **Password** and confirm your password.

Step 6 In the **User Group** area, click **Add** to add the **hetuuser** user group for the user.

NOTE

- Ranger authentication is enabled for the HetuEngine service in the MRS cluster by default. HetuEngine common users only need to be associated with the **hetuuser** user group. If Ranger authentication is disabled, you must associate the user with the **hive** user group and set it as the primary group. Otherwise, the HetuEngine service may be unavailable.
- If Ranger authentication is enabled and you need to configure the permissions to manage databases, tables, and columns of data sources for a user after it is created, see **Adding a Ranger Access Permission Policy for HetuEngine**.

Step 7 In the **Role** area, click **Add** to assign the **default** or desired tenant role permissions to the user.

Step 8 Click **OK**.

----End

11.2.3 Configuring Proxy User Authentication

This topic is available for MRS 3.3.0 or later.

You can use Ranger to authenticate a specified proxy user in HetuEngine for FusionInsight Manager user authentication. When you use the HetuEngine client, you can set **--session-user** to specify a proxy user.

For details about how to create an authentication user or proxy user, see **Creating a HetuEngine Permission Role**.

You need to enable Ranger authentication and grant the proxy user the permissions to manage the databases, tables, and columns of the data source. For details, see **Adding a Ranger Access Permission Policy for HetuEngine**.

- Kerberos authentication is enabled for the cluster (the cluster is in security mode)

  a. Use **kinit** to specify a user to be authenticated, for example, **hetuadmin1**. (The user must be a HetuEngine administrator and added to the **supergroup** user group to authenticate other users.)

     kinit hetuadmin1

     Enter the password as prompted and change the password upon your first login.

  b. Use **--session-user** to specify a proxy user, for example, **user1**.

     hetu-cli --session-user user1

- Kerberos authentication is disabled for the cluster (the cluster is in normal mode)

  Use **--user** to specify a user to be authenticated, for example, **user** (must belong to the **hetuuser** user group). Use **--session-user** to specify a proxy user, for example, **user1**.

  hetu-cli --user user --session-user user1

NOTE

This function is not suitable when both HiveMetastore data source authentication and multi-user mapping are enabled.

11.3 Quickly Using HetuEngine to Access Hive Data Source

This section describes how to use HetuEngine to connect to the Hive data source and query database tables of the Hive data source of the cluster through HetuEngine.

Prerequisites

- The HetuEngine and Hive services and their dependent services (DBService, KrbServer, ZooKeeper, HDFS, Yarn, and MapReduce) have been installed in the cluster and are running properly.
- If Kerberos authentication has been enabled for the cluster, you need to create a HetuEngine user and grant related permissions to the user in advance. For details, see **Creating a HetuEngine Permission Role**. In addition, you need to configure the permissions to manage the databases, tables, and columns of the data source for the user using Ranger. For details, see **Adding a Ranger Access Permission Policy for HetuEngine**.
- The cluster client has been installed in a directory, for example, **/opt/client**.

Procedure

Step 1 Create and start a HetuEngine compute instance.

1. Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.
2. In the **Basic Information** area on the **Dashboard** tab page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.
3. Click **Compute Instance** then **Create Configuration**.
   a. In the **Basic Configuration** area, set **Tenant** to the tenant associated with the user and configure **Instance Deployment Timeout Period (s)** and **Node Count**.
   b. Configure parameters in the **Coordinator Container Resource Configuration**, **Worker Container Resource Configuration**, and **Advanced Configuration** areas based on the actual resource plan. For details about the parameter configuration, see **Creating a HetuEngine Compute Instance** or retain the default values.

      NOTICE

      When you create a compute instance, you only need to apply for a few resources to test basic functions. You need to configure parameters based on actual service requirements and available resources. For details, see **Configuring HetuEngine Resource Groups** and **Configuring the Number of HetuEngine Worker Nodes**.

   c. Set **Start Now** to **Yes**, click **OK**, and wait until the instance configuration is complete.

Step 2 Log in to the node where the HetuEngine client is installed and run the following command to switch to the client installation directory:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Authenticate or specify a user.

- For a cluster in security mode, run the following command to authenticate the user:

  kinit *HetuEngine operation user*

  Example:

  kinit hetu_test

  Enter the password as prompted and change the password upon your first login.

- For a cluster in normal mode, run the following command to authenticate the user:

  hetu-cli --user *HetuEngine operation user*

  Example:

hetu-cli --user hetu_test

Step 5 Run the following command to log in to the catalog of the data source. You can choose to log in to the data source through ZooKeeper or HSFabric by configuring the **--mode** parameter.

- Through ZooKeeper (This mode is used by default if **--mode** is not configured.)

  hetu-cli --catalog *Data source name*

  For example, run the following command:

  hetu-cli --catalog hive

- Through HSFabric (Ensure that HSFabric instances exist.)

  hetu-cli --mode hsfabric --catalog *Data source name*

  For example, run the following command:

  hetu-cli --mode hsfabric --catalog hive

  NOTE

  - The default name of the Hive data source of the cluster is **hive**. To connect to data sources outside the cluster, configure external data sources on HSConsole by referring to **Adding a HetuEngine Data Source**.
  - It takes about 120 seconds for your first login to the client because the HetuEngine cluster needs to be started in the background.
  - Method to enable strict tenant verification for MRS 3.3.0 and later versions:

    HetuEngine provides service-level default resource queue configuration items. If no tenant information is specified, the default Yarn tenant is used. Multiple users may use the same queue by default.

    If you need to isolate resources and properly allocate SQL statements to specified queues, you can enable strict tenant verification by setting **tenant.strict.mode.enabled** to **true** and use the **--tenant** parameter to specify the queues when you use the client.

    Log in to Manager, choose **Cluster** > **Services** > **HetuEngine**, click **Configurations** and then **All Configurations**, search for **tenant.strict.mode.enabled**, set it to **true**, and save the settings. Click **Instance**, select all instances whose configurations expired, click **More**, select **Restart Instance**, and restart the instances as prompted for the configurations to take effect.
  - If strict tenant verification is enabled and cross-domain functions of HetuEngine are used, you need to set the **hsfabric.local.tenant** parameter of the HetuEngine data source. For details, see **Adding a Cross-Cluster HetuEngine Data Source**.

  Parameter description

  - **--mode**: Optional. The mode used to log in to a data source.
  - **--catalog**: Optional. The name of a specified data source.
  - **--tenant**: Optional. The tenant resource queue started by the cluster. If this parameter is not specified, the default tenant queue is used. To use this parameter, the service user must have the role permission of the tenant. For a cluster of MRS 3.3.0 or later versions:
    - This parameter is mandatory if strict verification is disabled.
    - This parameter is mandatory if strict verification is enabled.
  - **--schema**: Optional. The name of the schema of the data source to be accessed.
  - **--user**: Mandatory in normal mode. The name of the user who logs in to the client to execute services. The user must have at least the role of the queue specified by **--tenant** and cannot be an OS user.
  - You can run the **hetu-cli --help** command to view other parameters.

```
java -Djava.security.auth.login.config=/opt/client/HetuEngine/hetuserver/conf/jaas.conf -
Dzookeeper.sasl.clientconfig=Client -Dzookeeper.auth.type=kerberos -Djava.security.krb5.conf=/opt/client/
```



```
KrbClient/kerberos/var/krb5kdc/krb5.conf -Djava.util.logging.config.file=/opt/client/HetuEngine/hetuserver/
conf/hetuserver-client-logging.properties -jar /opt/client/HetuEngine/hetuserver/jars/hetu-cli-*-
executable.jar --catalog hive --deployment-mode on_yarn --server https://
10.112.17.189:24002,10.112.17.228:24002,10.112.17.150:24002?
serviceDiscoveryMode=zooKeeper&zooKeeperNamespace=hsbroker --krb5-remote-service-name HTTP --
krb5-config-path /opt/client/KrbClient/kerberos/var/krb5kdc/krb5.conf
hetuengine>
```

Step 6 Run the following command to view the database information:

show schemas;

```
    Schema
--------------------
 default
 information_schema
(2 rows)
Query 20230228_064136_00023_9kpap@default@HetuEngine, FINISHED, 3 nodes
Splits: 36 total, 36 done (100.00%)
0:02 [2 rows, 35B] [0 rows/s, 15B/s]
```

----End

11.4 Creating a HetuEngine Compute Instance

Scenario

This section describes how to create a HetuEngine compute instance. If you want to stop the cluster where compute instances are successfully created, you need to manually stop the compute instances first. If you want to use the compute instances after the cluster is restarted, you need to manually start them.

A single tenant can create multiple compute instances to balance loads, improving performance and fault tolerance (for MRS 3.3.0 and later versions only).

Prerequisites

- You have created a user for accessing the HetuEngine web UI, for example, **hetu_user**. For details, see **Creating a HetuEngine Permission Role**.
- You have created a tenant in the cluster to be operated. Ensure that the tenant has sufficient memory and CPUs when modifying the HetuEngine compute instance configuration.

  NOTE

  - You must use a leaf tenant when you create a HetuEngine compute instance because YARN jobs can only be submitted to the queues of a leaf tenant.
  - To avoid uncertainties caused by resource competition, you are advised to create independent resource pools for tenants used by HetuEngine.

- The startup of HetuEngine compute instances depends on Python 3. Ensure that Python 3 has been installed on all nodes in the cluster and the Python soft link has been added to the **/usr/bin/** directory. For details, see **Python Not Exist When a HetuEngine Compute Instance Failed to Start**.
- The HetuEngine service is running properly.

Procedure

Step 1 Log in to FusionInsight Manager as user **hetu_user** and choose **Cluster** > **Services** > **HetuEngine**.

Step 2 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 3 Click **Compute Instance** then **Create Configuration** and configure the compute instance parameters.

1. Set parameters in the **Basic Configuration** area. For details about the parameters, see **Table 11-7**.

Table 11-7 Basic configuration

Parameter	Description	Example Value
Tenant	Tenant to which the instance belongs. Only tenants without compute instances can be selected for new compute instances.	Select a value from the **Tenant** drop-down list.
Instance Deployment Timeout Period (s)	Timeout interval for starting a compute instance by Yarn service deployment. The system starts timing when the compute instance is started. If the compute instance is still in the **Creating** or **Starting** state after the time specified by this parameter expires, the compute instance status is displayed as **Error** and the compute instance that is being created or started on Yarn is stopped.	300 The value ranges from 1 to 2147483647.
Instance Count	The number of compute instances created under the current tenant. You can configure multiple compute instances for automatic load balancing and higher concurrency. **NOTICE** When you configure multiple compute instances: – Ensure that the cluster resources are sufficient. For multiple instances of a single tenant, required resources are single instance resources multiplied by the number of instances. The compute instance specifications must be the same. – Use short connections and use HSFabric for connecting HetuEngine, which enables better scheduling and a more balanced load.	1 Value range: 1 to 50

2. Set parameters in the **Coordinator Container Resource Configuration** area. For details about the parameters, see **Table 11-8**.

Table 11-8 Parameters for configuring Coordinator container resources

Parameter	Description	Example Value
Container Memory (MB)	Memory size (MB) allocated by Yarn to a single container of the compute instance Coordinator	Default value: 5120 The value ranges from 1 to 2147483647.
vcore	Number of vCPUs (vCores) allocated by Yarn to a single container of the compute instance Coordinator	Default value: 1 The value ranges from 1 to 2147483647.
Quantity	Number of containers allocated by Yarn to the compute instance Coordinator	Default value: 2 The value ranges from 1 to 3.
JVM	Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HetuEngine** > **Configurations**. On the **All Configurations** tab page, search for **extraJavaOptions**. The value of this parameter in the **coordinator.jvm.config** parameter file is the value of the JVM parameter.	-

3. Set parameters in the **Worker Container Resource Configuration** area. For details about the parameters, see **Table 11-9**.

Table 11-9 Parameters for configuring Worker container resources

Parameter	Description	Example Value
Container Memory (MB)	Memory size (MB) allocated by Yarn to a single container of the compute instance Worker	Default value: 10240 The value ranges from 1 to 2147483647.
vcore	Number of vCPUs (vCores) allocated by Yarn to a single container of the compute instance Worker	Default value: 1 The value ranges from 1 to 2147483647.
Quantity	Number of containers allocated by Yarn to the compute instance Worker	Default value: 2 The value ranges from 1 to 256.

Parameter	Description	Example Value
JVM	Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HetuEngine** > **Configurations**. On the **All Configurations** tab page, search for **extraJavaOptions**. The value of this parameter in the **worker.jvm.config** parameter file is the value of the JVM parameter.	-

4. Set parameters in the **Advanced Configuration** area. For details about the parameters, see **Table 11-10**.

Table 11-10 Advanced configuration parameters

Parameter	Description	Example Value
Ratio of Query Memory	This parameter indicates the ratio of the node query memory to the JVM memory and is set to 0.7 by default. When it is set to 0, the compute function is disabled. In this case, you can start the compute instance only if the value of -Xmx in the JVM configuration is no less than the sum of memory.heap-headroom-per-node and query.max-memory-per-node configured for the coordinator or worker.	0.7
Scaling	If auto scaling is enabled, you can increase or decrease the number of workers without restarting the instance. However, the instance performance may deteriorate. In multi-instance mode, automatic scaling cannot be enabled. For details about the parameters for enabling dynamic scaling, see **Configuring the Number of HetuEngine Worker Nodes**.	-
Maintenance Instance	To enable automatic refresh for materialized views, there must be one compute instance that is set as the maintenance instance and the compute instance is globally unique. If there are multiple compute instances, only one compute instance can be used as the maintenance instance.	-

5. Configure **Custom Configuration** parameters. You can add custom parameters to a specified parameter file. Select the specified parameter file from the **Parameter File** drop-down list.
   - You can click **Add** to add custom configuration parameters.
   - You can click **Delete** to delete custom configuration parameters.

- You can set **Parameter File** to **resource-groups.json** to configure the resource group mechanism. **Table 11-11** describe the resource group configuration parameter. For details about how to configure a resource group, see **Configuring HetuEngine Resource Groups**.

  Table 11-11 Resource group configuration parameter

Parameter	Description	Example Value

resourcegroups	Resource management group configuration of the cluster. Select **resource-groups.json** from the drop-down list of the parameter file.	{
		"rootGroups": [{
		"name": "global",
		"softMemoryLimit": "100%",
		"hardConcurrencyLimit": 1000,
		"maxQueued": 10000,
		"killPolicy": "no_kill"
		}],
		"selectors": [{
		"group": "global"
		}]
		}

  NOTE

  - After a custom parameter is configured in the **coordinator.config.properties**, **worker.config.properties**, **log.properties**, and **resource-groups.json** parameter files, if the parameter already exists in another specified parameter file, the value of this custom parameter will replace the values of this parameter in the specified parameter file. If the custom parameter does not exist in another specified parameter file, the custom parameter is added to the specified parameter file.
  - **killPolicy**: After a query is submitted to worker, if the total memory usage exceeds **softMemoryLimit**, you can select one of the following policies to terminate running queries:
    - **no_kill** (default value): Do not terminate the queries.
    - **recent_queries**: Terminate the queries based on the execution sequence in descending order.
    - **oldest_queries**: Terminate the queries based on the execution sequence.
    - **finish_percentage_queries**: Terminate the queries based on query execution percentage. The query with the smallest percentage of execution will be terminated first.
    - **high_memory_queries**: Terminate the queries based on memory usage. Queries with high memory usage are terminated first to free up more memory with the minimum number of query terminations. If the memory usage of two queries is less than 10%, the query with slower progress (smaller execution percentage) is terminated. If the difference between the execution percentages of two queries is less than 5%, the query with larger memory usage is terminated.

6. Determine whether to start the instance immediately after the configuration is complete.
   - If yes, the instance is automatically restarted immediately after the configuration is complete.
   - If no, you need to manually start the instance after the configuration is complete.

Step 4 Click **OK** and wait until the instance configuration is complete.

----End

Precautions for Maintaining Compute Instances

- During the restart or rolling restart of the HetuEngine service, do not create, start, stop, or delete HetuEngine compute instances on HSConsole.
- By default, a maximum of 10 compute instances can be in the starting, creating, deleting, stopping, scaling out, scaling in, or rolling restart state at the same time. O&M tasks that exceed 10 will wait to be executed in the background. To change the number of concurrent tasks, log in to FusionInsight Manager, choose **HetuEngine** and click the **Configurations** tab and then **All Configurations**. On the displayed page, search for **hsbroker.event.task.executor.threads** and change its value.
- Precautions for restarting HetuEngine compute instances
  - During the restart or rolling restart of HetuEngine compute instances, do not perform any change operations on the data sources on the HetuEngine and HetuEngine web UI, including restarting HetuEngine and changing its configurations.
  - If a compute instance has only one coordinator or worker node, do not perform a rolling restart of the instance.
  - If the number of worker nodes is greater than 10, the rolling restart of the instance may take more than 200 minutes. During this period, do not perform other O&M operations.
  - During the rolling restart of compute instances, HetuEngine releases YARN resources and applies for them again. Ensure that the CPU and memory of YARN are sufficient for starting 20% workers and YARN resources are not preempted by other jobs. Otherwise, the rolling restart will fail.

    Viewing YARN resources: Log in to FusionInsight Manager and choose **Tenant Resources**. On the navigation pane on the left, choose **Tenant Resources Management** to view the available queue resources of YARN in the **Resource Quota** area.

    Viewing the CPU and memory of a worker container: Log in to FusionInsight Manager as a user who can access the HetuEngine WebUI and choose **Cluster** > **Services** > **HetuEngine**. In the **Basic Information** area, click the link next to **HSConsole WebUI** to go to the HSConsole page. Click **Operation** in the row where the target instance is located and click **Configure**.
  - During the rolling restart, ensure that Application Manager of coordinators or workers in the YARN queue runs stably.
- HetuEngine compute instance restart exception handling
  - If Application Manager of coordinators or workers in the YARN queues is restarted during the rolling restart, the compute instances may be abnormal. In this case, you need to stop the compute instances and then start the compute instance for recovery.
  - After the rolling restart of a compute instance fails, the instance is in the subhealthy state. As a result, the configuration or number of coordinator or worker nodes may become inconsistent. In this case, the subhealthy

state of the compute instance cannot be automatically recovered. You need to manually check and rectify the fault, perform the rolling restart again, or stop and then restart the compute instance.

Compute Instance Statuses

After a compute instance is created, you can view information about the created instance on the **Compute Instance** tab page, including the tenant name, number of instances, instance status, and total resources. Instance statuses are as follows:

Figure 11-1 Compute instance statuses

- Green icon: The instance is in the running or subhealthy state.
- Red icon: The instance is faulty.
- Gray icon: The instance has been stopped and is to be started.
- Blue icon: The instance is in other states, including scaling out, scaling in, rolling restart, creating, starting, safely starting, shutting down, safely shutting down, terminating, terminated, and stopping.

11.5 Adding a HetuEngine Data Source

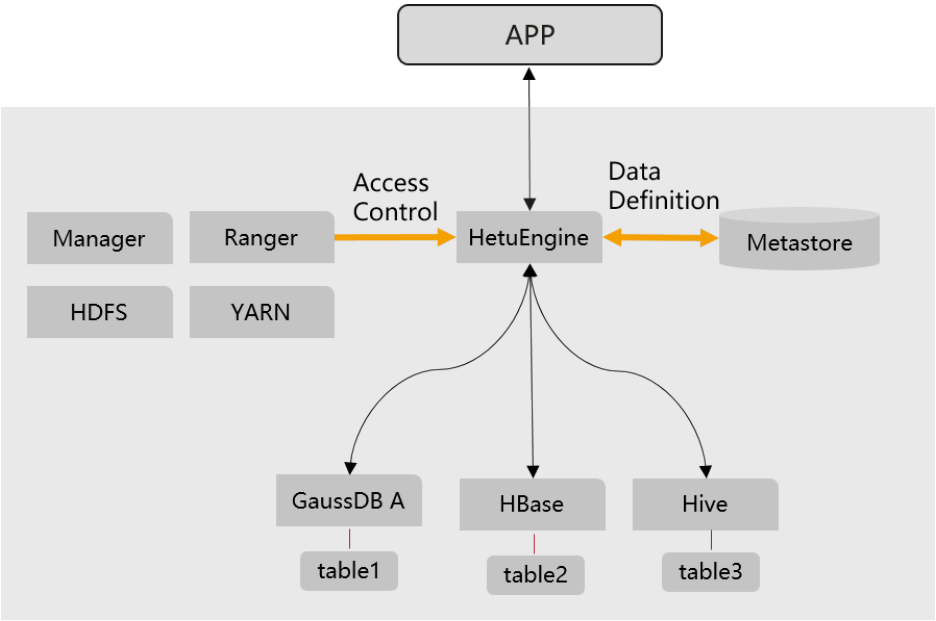
11.5.1 Using HetuEngine to Access Data Sources Across Sources and Domains

Using the HetuEngine Cross-Source Function

Companies often store large amounts of data from different sources in databases and warehouses to manage and collect information. However, having data from different sources with a variety of structures and scattered storage can make it expensive and time-consuming to query across all of them.

HetuEngine provides unified standard SQL statements to implement cross-source collaborative analysis, simplifying cross-source analysis operations.

Figure 11-2 HetuEngine cross-source analysis



Technologies and Advantages

- Compute pushdown: When HetuEngine is used for cross-source collaborative analysis, HetuEngine enhances the compute pushdown capability from the dimensions listed as follows to improve access efficiency.
  - Basic pushdown: predicate, projection, subquery, and limit
  - Aggregate pushdown: GROUP BY, ORDER BY, COUNT, SUM, MIN, and MAX
  - Operator pushdown: <, >, LIKE, and OR.
- Multi-source heterogeneous data: Collaborative analysis supports both structured data sources such as Hive, GaussDB, and ClickHouse, and unstructured data sources such as HBase and Elasticsearch.
- Global metadata: A mapping table is provided to map unstructured schemas to structured schemas, enabling HetuEngine to access HBase using SQL statements. Global management for data source information is provided.
- Global permission control: Data source permissions can be opened to Ranger through HetuEngine for centralized management and control.

Usage Guide of Cross-Source Analysis

HetuEngine supports quick joint query of multiple data sources and GUI-based data source configuration and management. You can quickly add the following data sources on the HSConsole page by referring to **Overview of HetuEngine Interactive Query**:

- **Adding a Hive Data Source**
- **Adding a Hudi Data Source**
- **Adding a ClickHouse Data Source**
- **Adding a GaussDB Data Source**

- **Adding an HBase Data Source**
- **Adding a Cross-Cluster HetuEngine Data Source**
- **Adding an IoTDB Data Source**
- **Adding a MySQL Data Source**
- **Adding an Oracle Data Source**
- **Adding a GBase Data Source**

Process of Using Cross-Source Collaborative Analysis

1. Log in to the HetuEngine client by referring to **Quickly Using HetuEngine to Access Hive Data Source**.
2. Register data sources such as Hive, HBase, and GaussDB A.

```
hetuengine> show catalogs;
 Catalog
----------
dws
hive
hive_dg
hbase
system
systemremote
(6 rows)
```

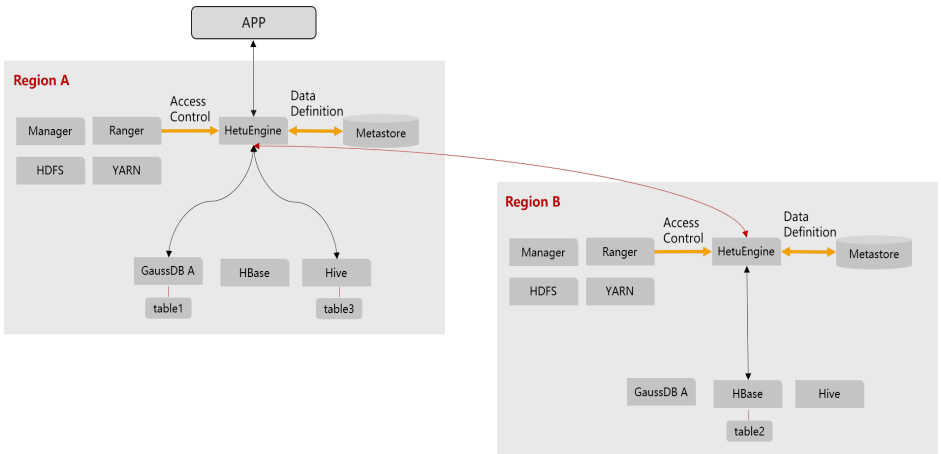
3. Compile SQL statements for cross-source collaborative analysis.

```
select * from hive_dg.schema1.table1 t1 join hbase.schema3.table3 t2 join dws.schema02.table4 t3 on t1.name = t2.item and t2.id = t3.cardNo;
```

HetuEngine Cross-Domain Function

HetuEngine provide unified standard SQL to implement efficient access to multiple data sources distributed in multiple regions (or data centers), shields data differences in the structure, storage, and region, and decouples data and applications.

Figure 11-3 HetuEngine cross-region functions



Key Technologies and Advantages

- No single-point failure bottleneck: HSFabric supports horizontal scale-out and multi-channel parallel transmission, maximizing the transmission rate. Cross-domain latency is no longer a bottleneck.

- Better computing resource utilization: Data compression and serialization tasks are delivered to Worker for parallel computing.
- Efficient serialization: The data serialization format is optimized to reduce the amount of data to be transmitted at the same data volume level.
- Streaming transmission: Based on HTTP 2.0 stream, ensure the universality of the HTTP protocol and reduce the repeated invoking of RPC during the transmission of a large amount of data.
- Resumable transmission: A large amount of data is prevented from being retransmitted after the connection is interrupted abnormally during data transmission.
- Traffic control: The network bandwidth occupied by data transmission can be limited by region to prevent other services from being affected due to exclusive traffic occupation in cross-region limited bandwidth scenarios.

Using a Cross-Domain Function

Prerequisites:

- At least one HSFabric instance has been deployed on the data nodes of the local and remote clusters.
- The nodes where the HSFabric instances of the local and remote clusters have been deployed can communicate with each other.

Procedure

Step 1 Open the data source in the local domain. You can create a virtual schema to shield the real schema information and instance information of the physical data source in the local domain from remote access requests. The remote end can use the virtual schema name to access the data source in the local domain.

```
CREATE VIRTUAL SCHEMA hive01.vschema01 WITH (
  catalog = 'hive01',
  schema = 'ins1'
);
```

Step 2 Register the data source of the HetuEngine type on the remote HetuEngine and add the local domain HetuEngine by referring to **Adding a Cross-Cluster HetuEngine Data Source**.

Step 3 Use cross-domain collaborative analysis.

```
// 1. Open the hive1.ins2 data source on the remote HetuEngine.
CREATE VIRTUAL SCHEMA hive1.vins2 WITH (
  catalog = 'hive1',
  schema = 'ins2'
);

// 2. Register three types of data sources, including Hive, GaussDB A, and HetuEngine, on HetuEngine in the local domain.
hetuengine> show catalogs;
  Catalog
-----------
dws
hetuengine_dc
hive
hive_dg
system
systemremote
(6 rows)

// 3. Perform cross-source collaborative analysis on HetuEngine in the local domain.
```



```
select * from hive_dg.schema1.table1 t1 join hetuengine_dc.vins2.table3 t2 join dws.schema02.table4 t3 on
t1.name = t2.item and t2.id = t3.cardNo;
```

----End

11.5.2 Adding a Hive Data Source

Introduction to Hive Data Sources

During HetuEngine installation, the co-deployed Hive data source (which is in the same cluster as HetuEngine) is interconnected by default. The data source name is **hive** and cannot be deleted. Some default configurations, such as the data source name, data source type, server principal, and client principal, cannot be modified. When the environment configuration changes, for example, the local domain name of the cluster is changed, restarting the HetuEngine service can automatically synchronize the configurations of the co-deployed Hive data source, such as server principal and client principal.

- Currently, HetuEngine supports data sources of the following data formats: AVRO, TEXT, RCTEXT, ORC, Parquet, and SequenceFile.
- When HetuEngine interconnects with Hive, you cannot specify multiple delimiters during table creation. However, if the MultiDelimitSerDe class is specified as the serialization class for a Hive data source to create a multi-delimiter table in text format, you can query the table using HetuEngine.
- The Hive data source interconnected with HetuEngine supports Hudi table redirection. This function is available to MRS 3.3.0 or later. Hudi table access requests are redirected to the Hudi connector, so the advanced functions of the Hudi connector are available. To use this function, you need to configure the target Hudi data source, ensure that the Metastore URL of the Hudi data source is the same as that of the current Hive data source, and enable Hudi redirection for the Hive data source.
- To use the isolation function of Hive Metastore, you need to configure **HIVE_METASTORE_URI_HETU** on Hive and restart the Hsbroke instance of the HetuEngine service to update the Hive Metastore URI.

Add a Hive data source outside a cluster on HSConsole.

Prerequisites for Adding a Hive Data Source

- The domain name of the cluster where the data source is located must be different from the HetuEngine cluster domain name.
- The cluster where the data source is located and the HetuEngine cluster nodes can communicate with each other.
- In the **/etc/hosts** file of all nodes in the cluster where HetuEngine is located, add the mapping between the host names and IP addresses of the cluster where the data source to be connected is located, and add **10.10.10.10 hadoop.***System domain name* in the **/etc/hosts** file (for example, **10.10.10.10 hadoop.hadoop.com**). Otherwise, HetuEngine cannot connect to the nodes that are not in the cluster based on the host name.
- A HetuEngine compute instance has been created.

Procedure for Adding a Hive Data Source

Step 1 Obtain the **hdfs-site.xml** and **core-site.xml** configuration files of the Hive data source cluster.

1. Log in to FusionInsight Manager of the cluster where the Hive data source is located.
2. In the upper right corner of the homepage, click **Download Client** to download the complete client to the local PC as prompted.
3. Decompress the downloaded client file package and obtain the **core-site.xml** and **hdfs-site.xml** files in the **FusionInsight_Cluster_1_Services_ClientConfig/HDFS/config** directory.
4. Check whether the **core-site.xml** file contains the **fs.trash.interval** configuration item. If not, add the following configuration items:

```
<property>
<name>fs.trash.interval</name>
<value>2880</value>
</property>
```

5. Change the value of **dfs.client.failover.proxy.provider.***NameService name* in the **hdfs-site.xml** file to **org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider**.

   For example, if the NameService name is **hacluster**, the configuration is as follows:

```
<property>
<name>dfs.client.failover.proxy.provider.hacluster</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
```

NOTICE

If the Hive data source to be interconnected is in the same Hadoop cluster with HetuEngine, you can log in to the HDFS client and run the following commands to obtain the **hdfs-site.xml** and **core-site.xml** configuration files. For details, see **Using the HDFS Client**.

hdfs dfs -get /user/hetuserver/fiber/restcatalog/hive/core-site.xml

hdfs dfs -get /user/hetuserver/fiber/restcatalog/hive/hdfs-site.xml

Step 2 Obtain the **user.keytab** and **krb5.conf** files of the proxy user of the Hive data source.

1. Log in to FusionInsight Manager of the cluster where the Hive data source is located.
2. Choose **System** > **Permission** > **User**.
3. Locate the row that contains the target data source user, click **More** in the **Operation** column, and select **Download Authentication Credential**.
4. Decompress the downloaded package to obtain the **user.keytab** and **krb5.conf** files.

   NOTE

   The proxy user of the Hive data source must be associated with at least the **hive** user group.

Step 3 Obtain the MetaStore URL and the Principal of the server.

1. Decompress the client package of the cluster where the Hive data source is located and obtain the **hive-site.xml** file from the **FusionInsight_Cluster_1_Services_ClientConfig/Hive/config** directory.
2. Open the **hive-site.xml** file and search for **hive.metastore.uris**. The value of **hive.metastore.uris** is the value of MetaStore URL. Search for **hive.server2.authentication.kerberos.principal**. The value of **hive.server2.authentication.kerberos.principal** is the value of Principal on the server.

Step 4 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 5 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 6 Choose **Data Source** and click **Add Data Source**. Configure parameters on the **Add Data Source** page.

1. In the **Basic Configuration** area, configure **Name** and choose **Hive** for **Data Source Type**.
2. Configure parameters in the **Hive Configuration** area. For details, see **Table 11-12**.

Table 11-12 Hive Configuration

Parameter	Description	Example Value
Driver	The default value is **fi-hive-hadoop**.	fi-hive-hadoop
hdfs-site File	Select the **hdfs-site.xml** configuration file obtained in **Step 1**. The file name is fixed.	-
core-site File	Select the **core-site.xml** configuration file obtained in **Step 1**. The file name is fixed.	-
yarn-site File	Obtain the file from the **Yarn/config** directory on the data source client. Upload this file only when the Hudi data source is connected.	-
krb5 File	Configure this parameter when the security mode is enabled. It is the configuration file used for Kerberos authentication. Select the **krb5.conf** file obtained in **Step 2**.	krb5.conf
Enable Data Source Authentication	Whether to use the permission policy of the Hive data source for authentication. If Ranger is disabled for the HetuEngine service, select **Yes**. If Ranger is enabled, select **No**.	No

3. In the **Metastore Configuration** area, configure the parameters according to **Table 11-13**.

Table 11-13 MetaStore Configuration

Parameter	Description	Example Value
Metastore URL	URL of the MetaStore of the data source. For details, see **Step 3**.	thrift://10.92.8.42:21088,thrift://10.92.8.43:21088,thrift://10.92.8.44:21088
Hudi Redirection (available for MRS 3.3.0 or later)	This parameter is available only when the Metastore URL of the target Hudi data source is the same as that of the current Hive data source. This function redirects Hudi table access request to the Hudi connector, so the advanced functions of the Hudi connector can be used.	No
Hudi Data Source (available for MRS 3.3.0 or later)	This parameter is required for Hudi redirection. All configured Hudi data sources are displayed in the drop-down list box. Select only the Hudi data source that has the same Metastore URL.	-
Security Authentication Mechanism	After the security mode is enabled, the default value is **KERBEROS**.	KERBEROS
Server Principal	Configure this parameter when the security mode is enabled. Value of **hive-site.xml** in **hive.server2.authentication.kerberos.principal** on the data source client. It specifies the username with domain name used by meta to access MetaStore. For details, see **Step 3**.	hive/hadoop.hadoop.com@HADOOP.COM

Parameter	Description	Example Value
Client Principal	Configure this parameter when the security mode is enabled. The parameter format is as follows: *username for accessing MetaStore*@*domain name (uppercase)*. *Username for accessing MetaStore* is the user to which the **user.keytab** file obtained in **Step 2** belongs. **NOTE** You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and view the value of **Local Domain**, which is the current system domain name, for example, HADOOP.COM.	admintest@HADOOP.COM
Keytab File	Configure this parameter when the security mode is enabled. It specifies the keytab credential file of the MetaStore user name. The file name is fixed. Select the **user.keytab** file obtained in **Step 2**.	user.keytab

4. Configure parameters in the **Connection Pool Configuration** area. For details, see **Table 11-14**.

   Table 11-14 Connection Pool Configuration

Parameter	Description	Example Value
Enable Connection Pool	Whether the connection pool is enabled when Hive MetaStore is accessed.	Yes
Maximum Connections	Maximum number of connections between a single Coordinator and Hive Metastore. Value range: 20 to 200; default value: 50	50

5. Configure parameters in **Hive User Information Configuration**. For details, see **Table 11-15**.

   Hive User Information Configuration and **HetuEngine-Hive User Mapping Configuration** must be used together. When HetuEngine is connected to the Hive data source, user mapping enables HetuEngine users to have the same permissions of the mapped Hive data source user. Multiple HetuEngine users can correspond to one Hive user.

Table 11-15 Hive User Information Configuration

Parameter	Description
Data Source User	Data source user information If the data source user is set to **hiveuser1**, a HetuEngine user mapped to **hiveuser1** must exist. For example, create **hetuuser1** and map it to **hiveuser1**.
Keytab File	Obtain the authentication credential of the user corresponding to the data source.

6. (Optional) Configure parameters in the **HetuEngine-Hive User Mapping Configuration** area. For details, see **Table 11-16**.

Table 11-16 HetuEngine-Hive User Mapping Configuration

Parameter	Description
HetuEngine User	HetuEngine user information.
Data Source User	Data source user information, for example, **hiveuser1** (data source user configured in **Table 11-15**)

7. (Optional) Modify custom configurations.
   - You can click **Add** to add custom configuration parameters by referring to **Table 11-17**.

Table 11-17 Custom parameters

Parameter	Description	Example Value
hive.metastore.connection.pool.maxTotal	Maximum number of connections in the connection pool.	50 (The value ranges from 20 to 200.)
hive.metastore.connection.pool.maxIdle	Maximum number of idle threads in the connection pool. When the number of idle threads reaches the maximum number, new threads are not released. Default value: **8**	8 (The value ranges from 0 to 200 and cannot exceed the maximum number of connections.)

Parameter	Description	Example Value
hive.metastore.connection.pool.minIdle	Minimum number of idle threads in the connection pool. When the number of idle threads reaches the minimum number, the thread pool does not create new threads. Default value: **0**	0 (The value ranges from 0 to 200 and cannot exceed the value of **hive.metastore.connection.pool.maxIdle**.)
hive.rcfile.time-zone	Adjusts the binary-encoded timestamp value to a specific time zone. When the table storage format is **RCBINARY** or **RCFILE**, the query result of timestamp data inserted by HetuEngine in Hive 3.1.0 or later is 8 hours earlier than that in HetuEngine. In this case, set this parameter to **UTC**. Default value: **JVM default** (obtaining the local time zone from JVM)	UTC
hive.orc.use-column-names	Whether to access ORC storage files by column name. The options are as follows: ■ **true**: yes ■ **false** (default value): no	false
hive.parquet.use-column-names	Whether to access Parquet storage files by column name. The options are as follows: ■ **true**: yes ■ **false** (default value): no	false
hive.hdfs.wire-encryption.enabled	This parameter needs to be added and set to **false** if the **hadoop.rpc.protection** parameter of the HDFS is set to **authentication** or **integrity**.	false

Parameter	Description	Example Value
hive.strict-mode-restrictions	You can configure the following constraints to restrict user query: ■ **NONE**: no constraints ■ **DISALLOW_EXCEEDED_SCAN_ON_PARTITION** (default value): The maximum number of partitions scanned in a single Hive partitioned table cannot be greater than the value of **hive.max-partitions-per-scan**.	DISALLOW_EXCEEDED_SCAN_ON_PARTITION
hive.ignore-absent-partitions	Query whether any file is missing in a partition. Value options are as follows: ■ **true**: Queries whether files are missing in the partition. ■ **false**: Do not query whether files are missing in the partition. In this case, an error is reported. If this parameter is left blank when data sources are manually connected, **false** is used by default.	true

– You can click **Delete** to delete custom configuration parameters.

NOTE

- You can prefix **coordinator.** or **worker.** to the custom parameters so that the parameters apply only to coordinator or worker nodes. For example, if you prefix **worker.** to **hive.metastore.connection.pool.maxTotal**, the custom parameter becomes **worker.hive.metastore.connection.pool.maxTotal**. If you set this new parameter to **50**, it indicates that a maximum number of 50 connections are allowed for worker nodes to access Hive MetaStore. If a custom parameter is not prefixed, the custom parameter is available for both coordinator and worker nodes.
- By default, the maximum number of connections for coordinator nodes to access Hive MetaStore is 50, and the maximum and minimum numbers of idle data source connections are 8 and 0, respectively. The maximum number of connections for worker nodes to access Hive MetaStore is 20, and the maximum and minimum numbers of idle data source connections are both 0.
- **hive.max-partitions-per-scan**: The maximum number of partitions scanned in a single Hive partitioned table. The default value is **100000**.
- The default value of **hive.ignore-absent-partitions** of the Hive data source co-deployed during HetuEngine installation is **true**.

8. Click **OK**.

Step 7 Log in to the node where the cluster client is located and run the following commands to switch to the client installation directory and authenticate the user:

cd /opt/client

source bigdata_env

kinit *User performing HetuEngine operations* (If the cluster is in normal mode, skip this step.)

Step 8 Run the following command to log in to the catalog of the data source:

hetu-cli --catalog *Data source name* **--schema** *Database name*

For example, run the following command:

hetu-cli --catalog hive_1 --schema default

Step 9 Run the following command. If the database table information can be viewed or no error is reported, the connection is successful.

show tables;

----End

Hive Data Type Mapping

Currently, Hive data sources support the following data types: BOOLEAN, TINYINT, SMALLINT, INT, BIGINT, REAL, DOUBLE, DECIMAL, NUMERIC, DEC, VARCHAR, VARCHAR (X), CHAR, CHAR (X), STRING, DATE, TIMESTAMP, TIME WITH TIMEZONE, TIMESTAMP WITH TIME ZONE, TIME, ARRAY, MAP, STRUCT, and ROW.

Performance Optimization

- Metadata caching

Hive connectors support metadata caching to provide metadata requests for various operations faster. For details, see **Adjusting HetuEngine Metadata Caching**.

- Dynamic filtering

  Enabling dynamic filtering helps optimize the calculation of the Join operator of Hive connectors. For details, see **Enabling Dynamic Filtering in HetuEngine**.

- Query with partitioning conditions

  Creating a partitioned table and querying data with partitioning filter criteria help filter out some partition data, improving performance.

- INSERT statement optimization

  You can improve insert performance by setting **task.writer-count** to **1** and choosing a larger value for **hive.max-partitions-per-writers**. For details, see **Optimizing HetuEngine INSERT Statements**.

Constraints on Hive Data Sources

- The DELETE syntax can be used to delete data from an entire table or a specified partition in a partitioned table.
- The Hive metabase does not support schema renaming, that is, the ALTER SCHEMA RENAME syntax is not supported.

11.5.3 Adding a Hudi Data Source

HetuEngine supports the query of COW/MOR table data. Configure a Hudi data source on HSConsole.

NOTE

HetuEngine cannot read Hudi bootstrap tables.

Prerequisites for Adding a Hudi Data Source

- You have created the proxy user of the Hudi data source. The proxy user is a human-machine user and must belong to the **hive** group.
- In the **/etc/hosts** file of all nodes in the cluster where HetuEngine is located, add the mapping between the host names and IP addresses of the cluster where the data source to be connected is located, and add **10.10.10.10 hadoop.***System domain name* in the **/etc/hosts** file (for example, **10.10.10.10 hadoop.hadoop.com**). Otherwise, HetuEngine cannot connect to the nodes that are not in the cluster based on the host name.
- You have created a HetuEngine administrator by referring to **Creating a HetuEngine Permission Role**.

Procedure for Adding a Hudi Data Source

Step 1 Obtain the **hdfs-site.xml** and **core-site.xml** configuration files of the Hudi data source cluster.

1. Log in to FusionInsight Manager of the cluster where the Hudi data source is.
2. In the upper right corner of the homepage, click **Download Client** to download the complete client to the local PC as prompted.

3. Decompress the downloaded client file package and obtain **core-site.xml** and **hdfs-site.xml** from the **FusionInsight_Cluster_1_Services_ClientConfig/HDFS/config** directory.
4. Check whether the **core-site.xml** file contains the **fs.trash.interval** configuration item. If not, add the following configuration items:

```
<property>
<name>fs.trash.interval</name>
<value>2880</value>
</property>
```

5. Change the value of **dfs.client.failover.proxy.provider.***NameService name* in the **hdfs-site.xml** file to **org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider**.

   For example, if the NameService name is **hacluster**, the configuration is as follows:

```
<property>
<name>dfs.client.failover.proxy.provider.hacluster</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
```

NOTICE

If the Hudi data source to be connected and HetuEngine are in the same Hadoop cluster, obtain the **hdfs-site.xml** and **core-site.xml** configuration files from the HDFS client of the cluster. To be specific, log in to the HDFS client by referring to **Using the HDFS Client** and run the following commands:

hdfs dfs -get /user/hetuserver/fiber/restcatalog/hive/core-site.xml

hdfs dfs -get /user/hetuserver/fiber/restcatalog/hive/hdfs-site.xml

Step 2 Obtain the **user.keytab** and **krb5.conf** files of the proxy user of the Hudi data source.

1. Log in to FusionInsight Manager of the cluster where the Hudi data source is.
2. Choose **System** > **Permission** > **User**.
3. Locate the row that contains the target data source user, click **More** in the **Operation** column, and select **Download Authentication Credential**.
4. Decompress the downloaded package to obtain **user.keytab** and **krb5.conf**.

   NOTE

   The proxy user of the Hive data source must be associated with at least the **hive** user group.

Step 3 Obtain the MetaStore URL and the Principal of the server.

1. Decompress the client package of the cluster where the Hudi data source is and obtain the **hive-site.xml** file from the **FusionInsight_Cluster_1_Services_ClientConfig/Hive/config** directory.
2. Open **hive-site.xml** and search for **hive.metastore.uris**. The value of **hive.metastore.uris** is the value of MetaStore URL. Search for **hive.server2.authentication.kerberos.principal**. The value of **hive.server2.authentication.kerberos.principal** is the value of Principal on the server.

Step 4 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 5 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 6 Choose **Data Source** and click **Add Data Source**. Configure parameters on the **Add Data Source** page.

1. In the **Basic Configuration** area, configure **Name** and choose **Hudi** for **Data Source Type**.
2. In the **Hudi Configuration** area, configure the parameters according to **Table 11-18**.

Table 11-18 Hudi configuration

Parameter	Description	Example Value
Driver	The default value is **hudi**.	hudi
hdfs-site File	Select the **hdfs-site.xml** configuration file obtained in **Step 1**. The file name is fixed.	-
core-site File	Select the **core-site.xml** configuration file obtained in **Step 1**. The file name is fixed.	-
krb5 File	Configure this parameter when the security mode is enabled. It is the configuration file used for Kerberos authentication. Select the **krb5.conf** file obtained in **Step 2**.	krb5.conf

3. In the **Metastore Configuration** area, configure the parameters according to **Table 11-19**.

Table 11-19 MetaStore Configuration

Parameter	Description	Example Value
Metastore URL	URL of the MetaStore of the data source. For details, see **Step 3**.	thrift://10.92.8.42:21088,thrift://10.92.8.43:21088,thrift://10.92.8.44:21088
Security Authentication Mechanism	After the security mode is enabled, the default value is **KERBEROS**.	KERBEROS

Parameter	Description	Example Value
Server Principal	Configure this parameter when the security mode is enabled. It specifies the username with domain name used by meta to access MetaStore. For details, see **Step 3**.	hive/hadoop.hadoop.com@HADOOP.COM
Client Principal	Configure this parameter when the security mode is enabled. The parameter format is as follows: *Username for accessing MetaStore*@*Domain name (uppercase)*.**COM**. *Username for accessing MetaStore* is the user to which the **user.keytab** file obtained in **Step 2** belongs.	admintest@HADOOP.COM
Keytab File	Configure this parameter when the security mode is enabled. It specifies the keytab credential file of the MetaStore username. The file name is fixed. Select the **user.keytab** file obtained in **Step 2**.	user.keytab

4. Click **OK**.

Step 7 Log in to the node where the cluster client is located and run the following commands to switch to the client installation directory and authenticate the user:

cd /opt/client

source bigdata_env

kinit *User performing HetuEngine operations* (If the cluster is in normal mode, skip this step.)

Step 8 Run the following command to log in to the catalog of the data source:

hetu-cli --catalog *Data source name* **--schema** *Database name*

For example, run the following command:

hetu-cli --catalog hudi --schema default

Step 9 Run the following command. If the database table information can be viewed or no error is reported, the connection is successful.

show tables;

----End

Hudi Data Type Mapping

Currently, the Hudi data source supports the following data types: INT, BIGINT, FLOAT, DOUBLE, DECIMAL, STRING, DATE, TIMESTAMP, BOOLEAN, BINARY, MAP, STRUCT and ARRAY.

Performance Optimization

- Metadata caching

  Hudi connectors support metadata caching to provide metadata requests for various operations faster. For details, see **Adjusting HetuEngine Metadata Caching**.

- Dynamic filtering

  Enabling dynamic filtering helps optimize the calculation of the Join operator of Hudi connectors. For details, see **Enabling Dynamic Filtering in HetuEngine**.

- Query with partition conditions

  Creating a partitioned table and querying data with partition filter criteria help filter out some partition data, improving performance.

Constraints on Hudi Data Source

Hudi data sources support only the QUERY operation.

11.5.4 Adding a ClickHouse Data Source

In a ClickHouse data source, a schema or database cannot contain tables with the same name but different case formats, for example, cktable (lowercase), CKTABLE (uppercase), and CKtable (uppercase and lowercase). Otherwise, HetuEngine cannot use the tables in the schema or database.

Procedure for Adding a ClickHouse Data Source

Step 1 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 2 In the **Basic Information** area on the **Dashboard** tab page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 3 Choose **Data Source** and click **Add Data Source**. On the **Add Data Source** page that is displayed, configure parameters.

1. In the **Basic Configuration** area, configure **Name** and choose **JDBC** > **ClickHouse** for **Data Source Type**.
2. Configure parameters in the **ClickHouse Configuration** area. For details, see **Table 11-20**.

Table 11-20 ClickHouse Configuration

Parameter	Description	Example Value
Driver	The default value is **clickhouse**.	clickhouse
JDBC URL	JDBC URL of the ClickHouse data source. – If the ClickHouse data source uses IPv4, the format is **jdbc:clickhouse://<host>:<port>**. – If the ClickHouse data source uses IPv6, the format is **jdbc:clickhouse://[<host>]:<port>**. – To obtain the value of **<host>**, log in to Manager of the cluster where the ClickHouse data source is located, choose **Cluster** > **Services** > **ClickHouse** > **Instance**, and view the ClickHouseBalancer service IP address. Select an IP address randomly. Currently, only one IP address can be entered. – To obtain the value of **<port>**, log in to Manager of the cluster where the ClickHouse data source is located, click **Cluster**, choose **Services** > **ClickHouse**, click **Configurations**, and click **All Configurations**. If the ClickHouse data source is in security mode, check the HTTPS port number of the ClickHouseBalancer instance, that is, the value of **lb_https_port**. If the ClickHouse data source is in normal mode, check the HTTP port number of the ClickHouseBalancer instance, that is, the value of **lb_http_port**.	**jdbc:clickhouse://10.162.156.243:21426** or **jdbc:clickhouse://10.162.156.243:21425**
Username	Username used for connecting to the ClickHouse data source.	Change the value based on the username being connected with the data source.
Password	User password used for connecting to the ClickHouse data source.	Change the value based on the user password for connecting to the data source.

Parameter	Description	Example Value
Case-sensitive Table/ Schema Name	Whether to support case-sensitive schema/ table names of the data source. HetuEngine supports case-sensitive schema/table names of the data source. – **No**: If multiple table names exist in the same schema of a data source, for example, **cktable** (lowercase), **CKTABLE** (uppercase), and **CKtable** (lowercase and uppercase), only **cktable** (lowercase) can be used by HetuEngine. – **Yes**: Only one table name can exist in the same schema of the data source, for example, **cktable** (lowercase), **CKTABLE** (uppercase), or **CKtable** (lowercase and uppercase). Otherwise, all tables in the schema cannot be used by HetuEngine.	-

3. (Optional) Customize the configuration.

   You can click **Add** to add custom configuration parameters. Configure custom parameters of the ClickHouse data source. For details, see **Table 11-21**.

Table 11-21 Custom parameters of the ClickHouse data source

Parameter	Description	Example Value
use-connection-pool	Whether to use the JDBC connection pool.	true
jdbc.connection.pool.maxTotal	Maximum number of connections in the JDBC connection pool.	8
jdbc.connection.pool.maxIdle	Maximum number of idle connections in the JDBC connection pool.	8
jdbc.connection.pool.minIdle	Minimum number of idle connections in the JDBC connection pool.	0
jdbc.connection.pool.testOnBorrow	Whether to check the connection validity when using a connection obtained from the JDBC connection pool.	false
clickhouse.map-string-as-varchar	Whether to convert the ClickHouse data source of the String and FixedString types to the Varchar type. Default value: **true**	true

Parameter	Description	Example Value
clickhouse.socket-timeout	Timeout interval for connecting to the ClickHouse data source. Unit: millisecond Default value: **120000**	120000
case-insensitive-name-matching.cache-ttl	Timeout interval for caching case-sensitive names of schemas or tables of the data sources. Unit: minute Default value: **1**	1

You can click **Delete** to delete custom configuration parameters.

4. Click **OK**.

Step 4 Log in to the node where the cluster client is located and run the following commands to switch to the client installation directory and authenticate the user:

cd /opt/client

source bigdata_env

kinit *User performing HetuEngine operations* (If the cluster is in normal mode, skip this step.)

Step 5 Run the following command to log in to the catalog of the data source:

hetu-cli --catalog *Data source name* **--schema** *Database name*

For example, run the following command:

hetu-cli --catalog clickhouse_1 --schema default

Step 6 Run the following command. If the database table information can be viewed or no error is reported, the connection is successful.

show tables;

----End

ClickHouse Data Type Mapping

Mapping from ClickHouse data types to HetuEngine data types

ClickHouse Data Type	HetuEngine Data Type
BOOLEAN	BOOLEAN
UInt8	SMALLINT
UInt16	INTEGER

ClickHouse Data Type	HetuEngine Data Type
UInt32	BIGINT
UInt64	DECIMAL(20, 0)
Int8	TINYINT
Int16	SMALLINT
Int32	INTEGER
Int64	BIGINT
Float32	REAL
Float64	DOUBLE
Decimal(P, S)	DECIMAL(P, S)
Decimal32(S)	DECIMAL(P, S)
Decimal64(S)	DECIMAL(P, S)
Decimal128(S)	DECIMAL(P, S)
IPv4	VARCHAR
IPv6	VARCHAR
UUID	VARCHAR
Enum8	VARCHAR
Enum16	VARCHAR
String	VARCHAR / VARBINARY
Fixedstring(N)	VARCHAR / VARBINARY
Date	DATE
DateTime	TIMESTAMP

Performance Optimization

- Subquery pushdown

  The query pushdown function is supported to improve query speed.
- Scalar UDF pushdown

  The Scalar UDF pushdown function is enabled by default. Before you use this function, create a mapping function in HetuEngine as needed.

Constraints on ClickHouse Data Source

- HetuEngine supports interconnecting with ClickHouse using the following SQL syntaxes: SHOW CATALOGS, SCHEMAS, TABLES, COLUMNS, DESCRIBE, USE, and SELECT TABLE/VIEW.

- Tables and views that support interconnection between HetuEngine and ClickHouse:

Item	Supported Table and View
Tables that support interconnection between HetuEngine and ClickHouse	Local table (MergeTree)
	Replicated table (ReplicatedReplacingMergeTree)
	Distributed table
Views that support interconnection between HetuEngine and ClickHouse	Normal view
	Materialized view

11.5.5 Adding a GaussDB Data Source

Add a GaussDB JDBC data source on HSConsole.

Prerequisites for Adding a GaussDB Data Source

- The cluster where the data source is located and the HetuEngine cluster nodes can communicate with each other.
- In the **/etc/hosts** file of all nodes in the cluster where HetuEngine is located, add the mapping between the host names and IP addresses of the cluster where the data source to be connected is located, and add **10.10.10.10 hadoop.***System domain name* in the **/etc/hosts** file (for example, **10.10.10.10 hadoop.hadoop.com**). Otherwise, HetuEngine cannot connect to the nodes that are not in the cluster based on the host name.
- A HetuEngine compute instance has been created.

Procedure for Adding a GaussDB Data Source

Step 1 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 2 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 3 Choose **Data Source** and click **Add Data Source**. Configure parameters on the **Add Data Source** page.

1. In the **Basic Configuration** area, configure **Name** and choose **JDBC** > **GAUSSDB-A** for **Data Source Type**.
2. Configure parameters in the **GAUSSDB-A Configuration** area. For details, see **Table 11-22**.

Table 11-22 GAUSSDB-A Configuration

Parameter	Description	Example Value
Driver	The default value is **gaussdba**.	gaussdba
JDBC URL	JDBC URL for connecting to GaussDB A. The format is as follows: **jdbc:postgresql://***CN service IP address***:***Port number***/***Database name* **NOTE** If SSL is enabled for the GaussDB database, add the following SSL-related parameter to the URL: **ssl=true&sslfactory=org.postgresql.ssl.NonValidatingFactory**	– SSL is disabled for the GaussDB database: jdbc:postgresql://10.0.136.1:25308/postgres – SSL is enabled for the GaussDB database: jdbc:postgresql://10.0.136.1:25308/postgres?ssl=true&sslfactory=org.postgresql.ssl.NonValidatingFactory
Username	Username for connecting to the GaussDB data source.	Change the value based on the username being connected with the data source.
Password	Password for connecting to the GaussDB data source.	Change the value based on the username and password for connecting to the data source.

3. (Optional) Configure GaussDB user information according to **Table 11-23**.

   GaussDB User Information Configuration and **HetuEngine-GaussDB User Mapping Configuration** must be used together. When HetuEngine is connected to the GaussDB data source, HetuEngine users can have the same permissions of the mapped GaussDB data source user through mapping. Multiple HetuEngine users can correspond to one GaussDB user.

   In the GaussDB database, the created user name must comply with the identifier naming convention and contain a maximum of 63 characters. If a username contains uppercase letters, the database automatically converts the uppercase letters into lowercase letters. To create a username that contains uppercase letters, enclose the username with double quotation marks (""). Therefore, you must use the converted username to set the **Data Source User** parameter.

   The examples are as follows:

   – If the user name created in the GaussDB database is **Gaussuser1**, the value of **Data Source User** must be **gaussuser1**.

- If the user name created in the GaussDB database is **"Gaussuser1"**, the value of **Data Source User** must be **Gaussuser1**.

Table 11-23 GaussDB User Information Configuration

Parameter	Description
Data Source User	Data source user name If the data source user is set to **gaussuser1**, a HetuEngine user mapped to **gaussuser1** must exist. For example, create **hetuuser1** and map it to **gaussuser1**.
Password	User authentication password of the corresponding data source

4. (Optional) Configure HetuEngine-GaussDB user mapping according to **Table 11-24**.

   Multiple HetuEngine accounts are configured in the format of **HetuEngine User** and **Data Source User** key-value pairs, corresponding to one of the users configured in the **GaussDB User Information Configuration** area. When different HetuEngine users are used to access GaussDB, different GaussDB usernames and passwords can be used.

   Table 11-24 HetuEngine-GaussDB User Mapping Configuration

Parameter	Description
HetuEngine User	HetuEngine username
Data Source User	Data source user, for example, **gaussuser1** (data source user configured in **Table 11-23**)

5. (Optional) Customize the configuration.
   - You can click **Add** to add custom configuration parameters. Configure custom parameters of the GaussDB data source. For details, see **Table 11-25**.

     Table 11-25 Custom parameters of the GaussDB data source

Parameter	Description	Example Value
use-connection-pool	Whether to use the JDBC connection pool.	true
jdbc.connection.pool.maxTotal	Maximum number of connections in the JDBC connection pool.	8
jdbc.connection.pool.maxIdle	Maximum number of idle connections in the JDBC connection pool.	8

Parameter	Description	Example Value
jdbc.connection.pool.minIdle	Minimum number of idle connections in the JDBC connection pool.	0
join-pushdown.enabled	▪ **true**: JOIN statements can be pushed down to the data source for execution. ▪ **false**: JOIN statements are not pushed down to the data source for execution. As a result, more network and compute resources are consumed.	true
join-pushdown.strategy	The JOIN push-down function should be enabled in advance. Value options are as follows: ▪ **AUTOMATIC**: cost-based JOIN pushdown ▪ **EAGER**: JOIN pushdown as much as possible	AUTOMATIC
source-encoding	GaussDB data source encoding mode.	UTF-8
multiple-cnn-enabled	Whether to use the GaussDB multi-CN configuration. To use it, ensure that the JDBC connection pool is disabled and the JDBC URL format is as follows: jdbc:postgresql://host:port/database,jdbc:postgresql://host:port/database,jdbc:postgresql://host:port/database.	false
parallel-read-enabled	Whether to use the parallel data read function. If the parallel data read function is enabled, the actual number of splits is determined based on the node distribution and the value of **max-splits**. Multiple connections to the data source will be created for parallel read operations. The dependent data source should support the load.	false

Parameter	Description	Example Value
split-type	Type of the parallel data read function. ■ **NODE**: The degree of parallelism (DOP) is categorized based on the GaussDB data source DataNodes. ■ **PARTITION**: The DOP is categorized based on table partitions. ■ **INDEX**: The DOP is categorized based on table indexes.	NODE
max-splits	Maximum degree of parallelism.	5
use-copymanager-for-insert	Whether to use CopyManager for batch import.	false
unsupported-type-handling	If the connector does not support the data of a certain type, convert it to VARCHAR. ■ After the **CONVERT_TO_VARCHAR** parameter is configured, the data of BIT VARYING, CIDR, MACADDR, INET, OID, REGTYPE, REGCONFIG and POINT types are converted to the varchar type during query and data of these types can only be read. ■ The default value is IGNORE, indicating that unsupported types will be not displayed in the result.	CONVERT_TO_VARCHAR
max-bytes-in-a-batch-for-copymanager-in-mb	Maximum volume of data imported by CopyManager in a batch, in MB.	10
decimal-mapping	By default, data of the DECIMAL, NUMBER, or NUMERIC type whose precision is not specified or exceeds the maximum precision of 38 digits is ignored. You can map the data to the DECIMAL(38, *x*) data type by setting the **decimal-mapping=allow_overflow** parameter. The value of *x* is the value of **decimal-default-scale**.	allow_overflow

Parameter	Description	Example Value
decimal-default-scale	Decimal precision when data of the DECIMAL, NUMBER, or NUMERIC type is mapped into DECIMAL(38, *x*). The value ranges from 0 to 38 and the default value is **0**.	0
case-insensitive-name-matching	HetuEngine supports case-sensitive table and schema names of the GaussDB data source. ■ **false**: Only schemas or tables whose names contain only lowercase letters can be queried. The default value is **false**. ■ **true**: If there are no schemas or tables with the same name after the case-insensitive matching, the schemas or tables can be queried. Otherwise, the schemas or tables cannot be queried.	false

  - You can click **Delete** to delete custom configuration parameters.

6. Click **OK**.

Step 4 Log in to the node where the cluster client is located and run the following commands to switch to the client installation directory and authenticate the user:

cd /opt/client

source bigdata_env

kinit *User performing HetuEngine operations* (If the cluster is in normal mode, skip this step.)

Step 5 Run the following command to log in to the catalog of the data source:

hetu-cli --catalog *Data source name* **--schema** *Database name*

For example, run the following command:

hetu-cli --catalog gaussdb_1 --schema admin

Step 6 Run the following command. If the database table information can be viewed or no error is reported, the connection is successful.

show tables;

----End

GAUSSDB Data Type Mapping

GaussDB Data Type	HetuEngine Data Type
BOOLEAN	BOOLEAN
TINYINT	TINYINT
SMALLINT	SMALLINT
INTEGER	INTEGER
BINARY_INTEGER	INTEGER
BIGINT	BIGINT
SMALLSERIAL	SMALLINT
SERIAL	INTEGER
BIGSERIAL	BIGINT
FLOAT4 (REAL)	REAL
FLOAT8(DOUBLE PRECISION)	DOUBLE PRECISION
DECIMAL[p (,s)]	DECIMAL[p (,s)]
NUMERIC[p (,s)]	DECIMAL[p (,s)]
CHAR(n)	CHAR(n)
CHARACTER(n)	CHAR(n)
NCHAR(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
CHARACTER VARYING(55)	VARCHAR(n)
VARCHAR2(n)	VARCHAR(n)
NVARCHAR2(n)	VARCHAR
TEXT(CLOB)	VARCHAR
DATE	TIMESTAMP
TIMESTAMP	TIMESTAMP
UUID	UUID
JSON	JSON

Constraints on GaussDB Data Source

- The following syntaxes are not supported: GRANT, REVOKE, SHOW GRANTS, SHOW ROLES, and SHOW ROLE GRANTS.

- The UPDATE and DELETE syntaxes cannot be used to filter clauses containing cross-catalog conditions, for example, **UPDATE mppdb.table SET column1=value WHERE column2 IN (SELECT column2 from hive.table)**.
- The UPDATE syntax cannot be used to update the DATE, TIMESTAMP, and VARBINARY fields.
- WHERE statements whose condition is REAL cannot be queried, for example, **SELECT * FROM mppdb.table WHERE column1 = REAL '1.1'**.
- The DELETE syntax cannot be used to filter clauses containing subqueries, for example, **DELETE FROM mppdb.table WHERE column IN (SELECT column FROM mppdb.table1)**.
- HetuEngine supports a maximum precision of 38 digits for GaussDB data sources, including the DECIMAL, NUMBER, and NUMERIC data types.
- If either end of a predicate contains a subquery, the predicate will not be pushed down. For example, if a subquery exists after the example statement **count(*)**, the predicate will not be pushed down, but the **min** function in the subquery can be pushed down.

```
select count(*) from item where i_current_price = (select min(i_current_price) from item);
```

11.5.6 Adding an HBase Data Source

This section describes how to add an HBase data source on HSConsole.

Prerequisites for Adding an HBase Data Source

- The domain name of the cluster where the data source is located must be different from the HetuEngine cluster domain name.
- The cluster where the data source is located and the HetuEngine cluster nodes can communicate with each other.
- In the **/etc/hosts** file of all nodes in the cluster where HetuEngine is located, add the mapping between the host names and IP addresses of the cluster where the data source to be connected is located, and add **10.10.10.10 hadoop.***System domain name* in the **/etc/hosts** file (for example, **10.10.10.10 hadoop.hadoop.com**). Otherwise, HetuEngine cannot connect to the nodes that are not in the cluster based on the host name.
- A HetuEngine compute instance has been created.
- The SSL communication encryption configuration of ZooKeeper in the cluster where the data source is located must be the same as that of ZooKeeper in the cluster where HetuEngine is located.

  NOTE

  To check whether SSL communication encryption is enabled, log in to FusionInsight Manager, choose **Cluster** > **Services** > **ZooKeeper** > **Configurations** > **All Configurations**, and enter **ssl.enabled** in the search box. If the value of **ssl.enabled** is **true**, SSL communication encryption is enabled. If the value is **false**, SSL communication encryption is disabled.

Procedure for Adding an HBase Data Source

Step 1 Obtain the **hbase-site.xml**, **hdfs-site.xml**, and **core-site.xml** configuration files of the HBase data source.

1. Log in to FusionInsight Manager of the cluster where the HBase data source is located.
2. In the upper right corner of the homepage, click **Download Client** to download the complete client as prompted.
3. Decompress the downloaded client file package and obtain the **hbase-site.xml**, **core-site.xml**, and **hdfs-site.xml** files in the **FusionInsight_Cluster_1_Services_ClientConfig/HBase/config** directory.

Step 2 Obtain the **user.keytab** and **krb5.conf** files of the proxy user of the HBase data source.

1. Log in to FusionInsight Manager of the cluster where the HBase data source is located.
2. Choose **System** > **Permission** > **User**.
3. Locate the row that contains the target data source user, click **More** in the **Operation** column, and select **Download Authentication Credential**.
4. Decompress the downloaded package to obtain the **user.keytab** and **krb5.conf** files.

NOTE

The proxy user of the data source must have the permission to perform HBase operations.

Step 3 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 4 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 5 Choose **Data Source** and click **Add Data Source**. Configure parameters on the **Add Data Source** page.

1. In the **Basic Configuration** area, configure **Name** and choose **HBase** for **Data Source Type**.
2. Configure parameters in the **HBase Configuration** area. For details, see **Table 11-26**.

Table 11-26 HBase Configuration

Parameter	Description	Example Value
Driver	The default value is **hbase-connector**.	hbase-connector

Parameter	Description	Example Value
ZooKeeper Quorum Address	Service IP addresses of all quorumpeer instances of the ZooKeeper service for the data source. If the ZooKeeper service of the data source uses IPv6, you need to specify the client port number in the ZooKeeper Quorum address. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ZooKeeper** > **Instance**, and view the IP addresses of all the hosts housing the quorumpeer instances.	– IPv4: 10.10.10.10,10.10.10.11,10.10.10.12 – IPv6: [10:10::10:11]:24002
ZooKeeper Client Port Number	Port number of the ZooKeeper client. Log in to FusionInsight Manager and choose **Cluster** > **Service** > **ZooKeeper**. On the **Configurations** tab page, check the value of **clientPort**.	2181
HBase RPC Communication Protection	Set this parameter based on the value of **hbase.rpc.protection** in the **hbase-site.xml** file obtained in **Step 1**. – If the value is **authentication**, set this parameter to **No**. – If the value is **privacy**, set this parameter to **Yes**.	No
Security Authentication Mechanism	After the security mode is enabled, the default value is **KERBEROS**.	KERBEROS
Principal	Configure this parameter when the security authentication mechanism is enabled. Set the parameter to the user to whom the **user.keytab** file obtained in **Step 2** belongs.	user_hbase@HADOOP2.COM
Keytab File	Configure this parameter when the security mode is enabled. It specifies the security authentication key. Select the **user.keytab** file obtained in **Step 2**.	user.keytab
krb5 File	Configure this parameter when the security mode is enabled. It is the configuration file used for Kerberos authentication. Select the **krb5.conf** file obtained in **Step 2**.	krb5.conf

Parameter	Description	Example Value
hbase-site File	Configure this parameter when the security mode is enabled. It is the configuration file required for connecting to HDFS. Select the **hbase-site.xml** file obtained in **Step 1**.	hbase-site.xml
core-site File	Configure this parameter when the security mode is enabled. This file is required for connecting to HDFS. Select the **core-site.xml** file obtained in **Step 1**.	core-site.xml
hdfs-site File	Configure this parameter when the security mode is enabled. This file is required for connecting to HDFS. Select the **hdfs-site.xml** file obtained in **Step 1**.	hdfs-site.xml

3. (Optional) Customize the configuration.
4. Click **OK**.

Step 6 Log in to the node where the cluster client is located and run the following commands to switch to the client installation directory and authenticate the user:

cd /opt/client

source bigdata_env

kinit *User performing HetuEngine operations* (If the cluster is in normal mode, skip this step.)

Step 7 Run the following command to log in to the catalog of the data source:

hetu-cli --catalog *Data source name* **--schema** *Database name*

For example, run the following command:

hetu-cli --catalog hbase_1 --schema default

Step 8 Run the following command. If the database table information can be viewed or no error is reported, the connection is successful.

show tables;

Step 9 Create a structured mapping table.

The format of the statement for creating a mapping table is as follows:

```
CREATE TABLE schemaName.tableName (
  rowId VARCHAR,
  qualifier1 TINYINT,
  qualifier2 SMALLINT,
  qualifier3 INTEGER,
  qualifier4 BIGINT,
  qualifier5 DOUBLE,
  qualifier6 BOOLEAN,
  qualifier7 TIME,
```



```
  qualifier8 DATE,
  qualifier9 TIMESTAMP
)
WITH (
column_mapping =
'qualifier1:f1:q1,qualifier2:f1:q2,qualifier3:f2:q3,qualifier4:f2:q4,qualifier5:f2:q5,qualifier6:f3:q1,qualifier7:f3:q2,
qualifier8:f3:q3,qualifier9:f3:q4',
row_id = 'rowId',
hbase_table_name = 'hbaseNamespace:hbaseTable',
external = true
);
```

NOTICE

The value of **schemaName** must be the same as that of **hbaseNamespace** in **hbase_table_name**.

- Supported mapping tables: Mapping tables can be directly associated with tables in the HBase data source or created and associated with new tables that do not exist in the HBase data source.
- Supported data types in a mapping table: VARCHAR, TINYINT, SMALLINT, INTEGER, BIGINT, DOUBLE, BOOLEAN, TIME, DATE, and TIMESTAMP
- The following table describes the keywords in the statements for creating mapping tables.

Table 11-27 Keywords in the statements for creating mapping tables

Keyword	Type	Mandatory	Default Value	Remarks
column_mapping	String	No	All columns belong to the same Family column family.	Specify the mapping between columns in the mapping table and column families in the HBase data source table. To associate a table in the HBase data source, set this parameter to the same value as that configured in the HBase data source. To create a table that does not exist in the HBase data source, configure this parameter. Value format: *Mapping table column name*:*HBase column family*.*HBase column name*. Mapping table column names must be in lowercase. HBase column names must be the same as that in HBase.

Keyword	Type	Mandatory	Default Value	Remarks
row_id	String	No	First column in the mapping table	Column name corresponding to the rowkey table in the HBase data source
hbase_table_name	String	No	N/A	Tablespace and table name of the HBase data source to be associated. Use a colon (:) to separate them. The default tablespace is **default**. If a new table that does not exist in the HBase data source is created, **hbase_table_name** does not need to be specified.
external	Boolean	No	true	If **external** is set to **true**, the table is a mapping table in the HBase data source and the original table in the HBase data source cannot be deleted. If **external** is set to false, the table in the HBase data source is deleted when the **Hetu-HBase** table is deleted.

----End

HBase Data Type Mapping

HBase is a byte-based distributed storage system that stores all data types as byte arrays. To represent HBase data in HetuEngine, select a data type that matches the value of the HBase column qualifier for the HetuEngine column qualifier by creating a mapping table in HetuEngine.

Currently, HetuEngine column qualifiers support the following data types: VARCHAR, TINYINT, SMALLINT, INTEGER, BIGINT, DOUBLE, BOOLEAN, TIME, DATE, and TIMESTAMP.

Performance Optimization

- Predicate pushdown

  Queries support pushdown of most operators. The following predicate conditions are supported: **=**, **>=**, **>**, **<**, **<=**, **!=**, **IN**, **NOT IN**, **IS NULL**, **IS NOT NULL**, and **BETWEEN AND**.

- Batch GET query

  Multiple row keys to be queried are encapsulated into one List<Get> in the HBase API, and then the list is requested to query data. In this way, each row key does not need to initiate a request separately.

- HBase single-table query range scanning optimization

  The HBase single-table query range scanning optimization is to automatically infer the start and end addresses of rowkeys based on the predicate conditions of HBase columns and configure the start and end addresses of HBase scan during tableScan for higher access performance.

  For example, assume that the rowkey of the HBase data table consists of four columns: **building_code**:**house_code**:**floor**:**uuid**. For the search criteria **where building_code = '123' and house_code = '456'**, the HetuEngine single-table query optimization scans only columns whose rowkey range prefixes are 123 to 456, improving performance.

  To enable the single HBase table query range scanning optimization function, add the custom parameter **hbase.rowkey.adaptive.optimization.enabled** to **5.3** and set it to **true**.

  In addition, you need to specify the columns and separators of rowkeys in the table creation property of table creation statements.

  Table 11-28 Columns and separators of HBase rowkeys

Table Property	Description	Example Value
row_id_construct_columns	Columns of rowkeys in an HBase data table	building_code:house_code:floor:uuid
row_id_construct_columns_terminal	Separator of columns of rowkeys in an HBase data table	:

  For example, a table creation statement containing a rowkey consisting of four columns building_code:house_code:floor:uuid is as follows:

```
CREATE TABLE test.table_hbase_test (
row_id string,
col1 string,
col2 string,
col3 string,
building_code string,
house_code string,
floor string,
uuid string)
WITH (column_mapping = '
col1:attr:col1,
col2:attr:col2,
col3:attr:col3,
building_code:attr:building_code,
house_code:attr:house_code,
floor:attr:floor,
uuid:attr:uuid',
row_id = 'row_id',
row_id_construct_columns = 'building_code:house_code:floor:uuid',
row_id_construct_columns_terminal = ':',
hbase_table_name='test:table_hbase_test',
external = true)
```

- Dynamic filtering optimization for HBase multi-table join query

  HBase supports dynamic filtering optimization.

  To enable the dynamic filtering function, enable the HBase single table query range scanning optimization function, add the custom parameter **enable-**

dynamic-filtering in the **coordinator.config.properties** and **worker.config.properties** parameter files of compute instances, and set the parameter to **true**. For details, see **Step 3.5**.

Constraints on HBase Data Source

The ALTER and VIEW syntaxes are not supported.

11.5.7 Adding a Cross-Cluster HetuEngine Data Source

This section describes how to add another HetuEngine data source on the HSConsole page for a cluster in security mode.

Procedure for Adding a Cross-Cluster HetuEngine Data Source

Step 1 Obtain the **user.keytab** file of the proxy user of the HetuEngine cluster in a remote domain.

1. Log in to FusionInsight Manager of the HetuEngine cluster in the remote domain.
2. Choose **System** > **Permission** > **User**.
3. Locate the row that contains the target data source user, click **More** in the **Operation** column, and select **Download Authentication Credential**.
4. The **user.keytab** file extracted from the downloaded file is the user credential file.

Step 2 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 3 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 4 Choose **Data Source** and click **Add Data Source**. Configure parameters on the **Add Data Source** page.

1. In the **Basic Configuration** area, configure **Name** and choose **HetuEngine** for **Data Source Type**.
2. Configure parameters in the **HetuEngine Configuration** area. For details, see **Table 11-29**.

Table 11-29 HetuEngine Configuration

Parameter	Description	Example Value
Driver	The default value is **hsfabric-initial**.	hsfabric-initial
Username	Configure this parameter when the security mode is enabled. It specifies the user who accesses the remote HetuEngine. Set the parameter to the user to whom the **user.keytab** file obtained in **Step 1** belongs.	hetu_test

Parameter	Description	Example Value
Keytab File	Configure this parameter when the security mode is enabled. This is the Keytab file of the user who accesses the remote DataCenter. Select the **user.keytab** file obtained in **Step 1**.	user.keytab
Two Way Transmission	This parameter indicates whether to enable bidirectional transmission for cross-domain data transmission. The default value is **Yes**. – **Yes**: Two-way transmission: Requests are forwarded to the remote HSFabric through the local HSFabric. If two-way transmission is enabled, the local HSFabric address must be configured. – No: Unidirectional transmission: Requests are directly sent to the remote HSFabric.	Yes
Local Configuration	Host IP address and port number of the HSFabric instance that is responsible for external communication of the HetuEngine service in the local MRS cluster. 1. Log in to FusionInsight Manager of the local cluster, choose **Cluster** > **Services** > **HetuEngine** > **Instance**, and check the service IP address of the HSFabric. 2. Click **HSFabric**, choose **Instance Configuration**, and check the value of **server.port**. The default value is **29900**.	192.162.157.32:29900
Remote Address	Host IP address and port number of the HSFabric instance that is responsible for external communication of the HetuEngine service in the remote MRS cluster. 1. Log in to FusionInsight Manager of the remote cluster, choose **Cluster** > **Services** > **HetuEngine** > **Instance**, and check the service IP address of the HSFabric. 2. Click **HSFabric**, choose **Instance Configuration**, and check the value of **server.port**. The default value is **29900**.	192.168.1.1:29900

Parameter	Description	Example Value
Region	Region to which the current request initiator belongs. The value can contain only digits and underscores (_).	0755_01
Receiving Data Timeout (s)	Timeout interval for receiving data, in seconds.	60
Total Task Timeout (s)	Total timeout duration for executing a cross-domain task, in seconds.	300
Tasks Used by Worker Nodes	Number of tasks used by each worker node to receive data.	5
Data Compression	– **Yes**: Data compression is enabled. – **No**: Data compression is disabled.	Yes

3. (Optional) Customize the configuration.
   – You can click **Add** to add custom configuration parameters. Configure custom parameters of the HetuEngine data source. For details, see **Table 11-30**.

Table 11-30 Custom parameters of the HetuEngine data source

Parameter	Description	Example Value
hsfabric.health.check.time	Interval for checking the HSFabric instance status, in seconds.	60
hsfabric.subquery.pushdown	Whether to enable cross-domain query pushdown. The function is enabled by default. ▪ **true**: enables cross-domain query pushdown. ▪ **false**: disables cross-domain query pushdown.	true

Parameter	Description	Example Value
hsfabric.local.tenant (available for MRS 3.3.0 or later)	Tenant queue used by the remote HetuEngine for computing ■ If this parameter is not set, the system randomly selects the tenant to which the user belongs based on the configured user. ■ If this parameter is set, the specified tenant will be used. This parameter applies to scenarios where strict tenant verification is enabled.	-

  - You can click **Delete** to delete custom configuration parameters.

4. Click **OK**.

Step 5 Log in to the node where the cluster client is located and run the following commands to switch to the client installation directory and authenticate the user:

cd /opt/client

source bigdata_env

kinit *User performing HetuEngine operations* (If the cluster is in normal mode, skip this step.)

Step 6 Run the following command to log in to the catalog of the data source:

hetu-cli --catalog *Data source name* **--schema** *Database name*

For example, run the following command:

hetu-cli --catalog hetuengine_1 --schema default

Step 7 Run the following command. If the database table information can be viewed or no error is reported, the connection is successful.

show tables;

----End

Cross-Cluster HetuEngine Data Type Mapping

Currently, HetuEngine data sources support the following data types: BOOLEAN, TINYINT, SMALLINT, INT, BIGINT, REAL, DOUBLE, DECIMAL, VARCHAR, CHAR, DATE, TIMESTAMP, ARRAY, MAP, TIME WITH TIMEZONE, TIMESTAMP WITH TIME ZONE, and TIME.

Performance Optimization

The query pushdown function is supported to improve query speed.

This function is enabled by default. You can also enable it by adding related custom parameters according to **Step 4.3**.

Constraints on Cross-Cluster HetuEngine Data Source

The following syntaxes are not supported: CREATE, ALTER, DROP VIEW, INSERT OVERWRITE, UPDATE, and DELETE.

INSERT is not supported for cross-domain data sources.

11.5.8 Adding an IoTDB Data Source

This section applies to MRS 3.2.0 or later.

Add an IoTDB JDBC data source on HSConsole of a cluster in security mode.

Prerequisites for Adding an IoTDB Data Source

- The domain name of the cluster where the data source is located must be different from that of the HetuEngine cluster.
- The cluster where the data source is located and the HetuEngine cluster nodes can communicate with each other.
- A HetuEngine compute instance has been created.
- By default, SSL is enabled for the IoTDB service in a security cluster. After SSL is enabled, you need to upload the **truststore.jks** file. For details about how to obtain the file, see **Using the IoTDB Client**.

Procedure for Adding an IoTDB Data Source

Step 1 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**.

Step 2 On the **Dashboard** tab page that is displayed, find the **Basic Information** area, and click the link next to **HSConsole WebUI**.

Step 3 Choose **Data Source** and click **Add Data Source**. Configure parameters on the **Add Data Source** page.

1. In the **Basic Configuration** area, configure **Name** and choose **JDBC** > **IoTDB** for **Data Source Type**.
2. Configure parameters in the **IoTDB Configuration** area by referring to **Table 11-31**.

Table 11-31 IoTDB configuration parameters

Parameter	Description	Example Value
Driver	The default value is **iotdb**.	iotdb

Parameter	Description	Example Value
JDBC URL	JDBC URL for connecting to IoTDB. – If the IoTDB data source uses an IPv4 address, the format is **jdbc:iotdb://** *P address 1 of the IoTDBServer service, IP address 2 of the IoTDBServer service*:*Port*. – If the IoTDB data source uses an IPv6 address, the format is **jdbc:iotdb://[** *P address 1 of the IoTDBServer service, IP address 2 of the IoTDBServer service*]:*Port*.	– IPv4: **jdbc:iotdb://10.10.10.11,10.10.10.12:22260** – IPv6: **jdbc:iotdb://[10:10::10:11,10:10::10:12]:22260**
Username	IoTDB username for connecting to the IoTDB data source	**NOTE** If the cluster where IoTDB resides is in non-security mode, set this parameter to the default IoTDB user **root**.
Password	Password of the IoTDB username for connecting to the IoTDB data source	**NOTE** If the cluster where the IoTDB service is installed is in non-security mode, obtain the password of user **root** from the administrator of this cluster.
Enable SSL	Whether SSL is enabled for the IoTDB service. SSL is enabled by default in a security cluster.	Yes
truststore File	After SSL is enabled for IoTDB, upload the **truststore.jks** file.	-

NOTE

- Service IP addresses of IoTDBServer:

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **IoTDB**. On the page that is displayed, click the **Instance** tab. On this tab page, check **Service IP Address** of IoTDBServer.

- Port number:

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **IoTDB**. On the page that is displayed, click the **Configurations** tab. On this tab page, search for and check the value of **IOTDB_SERVER_RPC_PORT**. The default value is **22260**.

3. (Optional) Add custom configurations as needed.

4. Click **OK**.

Step 4 Log in to the node where the cluster client is located and run the following commands to switch to the client installation directory and authenticate the user:

cd /opt/client

source bigdata_env

kinit *User performing HetuEngine operations* (If the cluster is in normal mode, skip this step.)

Step 5 Run the following command to log in to the catalog of the data source:

hetu-cli --catalog *Data source name* **--schema** *Database name*

For example, run the following command:

hetu-cli --catalog iotdb_1 --schema root.ln

Step 6 Run the following command. If the database table information can be viewed or no error is reported, the connection is successful.

show tables;

----End

IoTDB Data Type Mapping

IoTDB Data Type	HetuEngine Data Type
BOOLEAN	BOOLEAN
INT32	BIGINT
INT64	BIGINT
FLOAT	DOUBLE
DOUBLE	DOUBLE
TEXT	VARCHAR

Function Enhancement

- IoTDB can configure any label fields for time series. These IoTDB label fields and other data sources can be jointly queried through HetuEngine.
- Any nodes from the IoTDB database level to the time series can be used as tables for data query on HetuEngine.

Constraints on IoTDB Data Source

- IoTDB data cannot be created but can be queried.
- The IoTDB user who uses HetuEngine for query must at least be configured with the read permission on the root directory.

11.5.9 Adding a MySQL Data Source

This section applies to MRS 3.3.0 or later.

You can interconnect HetuEngine with MySQL data sources to access and query MySQL data. This section describes how to add a MySQL JDBC data source on HSConsole.

Prerequisites for Adding a MySQL Data Source

- The data source and the HetuEngine cluster nodes can communicate with each other.
- If Kerberos authentication is enabled for the cluster (the cluster is in security mode), create a HetuEngine administrator user. If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), create a HetuEngine service user, and assign the HDFS administrator permission to the user. That is, the user is added to both the **hadoop** and **hadoopmanager** user groups. For details about how to create a user, see **Creating a HetuEngine Permission Role**.
- A HetuEngine compute instance has been created. For details, see **Creating a HetuEngine Compute Instance**.
- You have obtained the IP address, port number, username, and password for logging in to the MySQL database.

Constraints on Interconnection Between HetuEngine and MySQL Data Sources

- HetuEngine supports interconnecting with MySQL using the following SQL syntaxes: SHOW CATALOGS, SCHEMAS, TABLES, COLUMNS, DESCRIBE, USE, and SELECT TABLE/VIEW.
- The schema and table names of MySQL data sources supported by HetuEngine are case insensitive.
- Predicate pushups or pushdowns are not allowed on columns of text types such as CHAR or VARCHAR.

  For example, if **name** is a column of the VARCHAR type, the predicates of the following two queries cannot be pushed down.

  ```
  SELECT * FROM nation WHERE name>'abcd';
  SELECT * FROM nation WHERE name='abcd';
  ```

Configuring a MySQL Data Source

Installing a cluster client

Step 1 Install the cluster client that contains the HetuEngine service in the **/opt/hadoopclient** directory.

Prepare the MySQL driver

Step 2 Obtain the MySQL driver file (*xxx*.jar) from the MySQL official website. The supported versions are MySQL 5.7, MySQL 8.0, and later versions.

Step 3 Upload the MySQL driver file to the cluster where HetuEngine is deployed.

You can use either of the following methods:

- Upload the file to HDFS on FusionInsight Manager.
  a. Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HDFS**.
  b. In the **Basic Information** area on the **Dashboard** page, click the link next to **NameNode Web UI**.
  c. Select **Utilities** > **Browse the file system**, click  , and create the **/user/hetuserver/fiber/extra_file/driver/mysql** directory.
  d. Go to the **/user/hetuserver/fiber/extra_file/driver/mysql** directory and click  to upload the MySQL driver file obtained in **Step 2**.
  e. Click the value in the **Permission** column in the row containing the driver file, select **Read** and **Write** in the **User** column, **Read** in the **Group** column, and **Read** in the **Other** column, and click **Set**.
- Run HDFS commands to upload the file.
  a. Log in to the node where the HDFS service client is deployed and switch to the client installation directory, for example, **/opt/hadoopclient**.

     cd /opt/hadoopclient
  b. Configure environment variables.

     source bigdata_env
  c. If the cluster is in security mode, authenticate the user. For a normal cluster, user authentication is not required.

     kinit *HetuEngine administrator username*

     Enter the password as prompted.
  d. Run the following commands to create **/user/hetuserver/fiber/extra_file/driver/mysql**, upload the MySQL driver obtained in **Step 2**, and modify the permission:

     hdfs dfs -mkdir -p /user/hetuserver/fiber/extra_file/driver/mysql

     hdfs dfs -put ./*MySQL driver file* **/user/hetuserver/fiber/extra_file/driver/mysql**

     hdfs dfs -chmod -R 644 /user/hetuserver/fiber/extra_file/driver/mysql

Configuring a MySQL Data Source

Step 4 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**.

Step 5 In the displayed**Dashboard** tab, find the **Basic Information** area, and click the link next to **HSConsole WebUI**.

Step 6 Choose **Data Source** and click **Add Data Source**. Configure parameters on the **Add Data Source** page.

1. In the **Basic Configuration** area, configure **Name** and choose **JDBC** > **MySQL** for **Data Source Type**.
2. In the **MySQL Configuration** area, configure the parameters according to **Table 11-32**.

Table 11-32 MySQL configuration

Parameter	Description	Example Value
Driver	The default value is **mysql**.	mysql
Driver Name	Select the MySQL driver that has been uploaded in **Step 2**. The driver format is *xxx*.jar.	mysql-connector-java-8.0.11.jar
JDBC URL	JDBC URL for connecting to MySQL. Format: **jdbc:mysql://***IP address of the MySQL database*:*Port number*. The default port number is **3306**.	– IPV4: jdbc:mysql://10.10.10.11:3306 – IPV6: jdbc:mysql://[10:10::10:11]:3306
Username	MySQL username for connecting to the MySQL data source	-
Password	Password of the MySQL username for connecting to the MySQL data source	-

3. (Optional) Customize the configuration.

   Click **Add** to add custom configuration parameters. Configure custom parameters of the MySQL data source. For details, see **Table 11-33**.

Table 11-33 Custom parameters of the MySQL data source

Parameter	Description	Example Value
mysql.auto-reconnect	Whether to reconnect automatically – **true** (default value): Enable automatic reconnection. – **false**: Disable automatic reconnection.	true
mysql.max-reconnects	Maximum number of reconnection attempts. The default value is 3.	3
mysql.jdbc.use-information-schema	Whether the driver should use INFORMATION_SCHEMA to derive the information used by DatabaseMetaData.	true
use-connection-pool	Whether to use the JDBC connection pool. The default value is **false**.	false
jdbc.connection.pool.maxTotal	Maximum number of connections in the JDBC connection pool. The default value is **8**.	8

Parameter	Description	Example Value
jdbc.connection.pool.maxIdle	Maximum number of idle connections in the JDBC connection pool. The default value is **8**.	8
jdbc.connection.pool.minIdle	Minimum number of idle connections in the JDBC connection pool. The default value is **0**.	0
case-insensitive-name-matching	The schema and table names of MySQL data sources supported by HetuEngine are case sensitive. – **false** (default value): Only schemas and tables whose names contain only lowercase letters can be queried. – **true**: ▪ If no schema or table is matched ignoring case sensitivity, the schema and table can be queried. ▪ If schemas and tables are matched ignoring case sensitivity, the schema and table cannot be queried.	false
case-insensitive-name-matching.cache-ttl	Timeout interval for caching case-sensitive schema and table names of the MySQL data source. The default value is 1 minute.	1m
dynamic-filtering.enabled	Whether dynamic filters will be pushed down to JDBC queries. – **true** (default value): Enable pushdown. – **false**: Disable pushdown.	true
dynamic-filtering.wait-timeout	The maximum duration that HetuEngine will wait to collect dynamic filters from the build side of the connection before starting a JDBC query. Using a larger value may result in a more detailed dynamic filter. However, the latency of some queries is increased. The default value is 20s.	20s
unsupported-type-handling	How data types that are not supported by the connector will be processed – **CONVERT_TO_VARCHAR**: Convert unsupported types to **VARCHAR** and allow only read operations on them. – **IGNORE** (default value): Do not display the unsupported types.	IGNORE

Parameter	Description	Example Value
join-pushdown.enabled	Whether join pushdown is enabled. – **true** (default value): Enable join pushdown. – **false**: Disable join pushdown.	true
join-pushdown.strategy	Policy used to evaluate whether the Join operation is pushed down. – **AUTOMATIC** (default value): Enable cost-based connection pushdown. – **EAGER**: Push down joins as much as possible. Even if table statistics are unavailable, using **EAGER** will push down joins, which may cause query performance deterioration. Use **EAGER** only in test and troubleshooting scenarios.	AUTOMATIC

Click **Delete** to delete custom configuration parameters.

4. Click **OK**

Step 7 Log in to the node where the cluster client is deployed and run the following commands to switch to the client installation directory and authenticate the user:

cd /opt/hadoopclient

source bigdata_env

kinit *User performing HetuEngine operations* (If the cluster is in normal mode, skip this command.)

Step 8 Log in to the catalog of the data source.

hetu-cli --catalog *Data source name* **--schema** *Database name*

For example, run the following command:

hetu-cli --catalog mysql_1 --schema mysql

Step 9 Run the following command. If the database table information can be viewed or no error is reported, the connection is successful.

show tables;

----End

Mapping Between MySQL and HetuEngine Data Types

Mapping from MySQL data types to HetuEngine data types

MySQL Type	HetuEngine Data Type
BIT	BOOLEAN
BOOLEAN	TINYINT
TINYINT	TINYINT
SMALLINT	SMALLINT
INTEGER	INTEGER
BIGINT	BIGINT
DOUBLE PRECISION	DOUBLE
FLOAT	REAL
REAL(m, d)	REAL(m, d)
DECIMAL(p, s)	DECIMAL(p, s)
CHAR(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
TINYTEXT	VARCHAR(255)
TEXT	VARCHAR(65535)
MEDIUMTEXT	VARCHAR(16777215)
LONGTEXT	VARCHAR
ENUM(n)	VARCHAR(n)
BINARY, VARBINARY, TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB	VARBINARY
JSON	JSON
DATE	DATE
TIME(n)	TIME(n)
DATETIME(n)	TIMESTAMP(n)
TIMESTAMP(n)	TIMESTAMP(n)

11.5.10 Adding an Oracle Data Source

This topic is available for MRS 3.5.0 and later versions only.

HetuEngine allows you to configure, access, and query the Oracle data source. This topic guides you to add an Oracle JDBC data source on the HSConsole page of the cluster.

Prerequisites

- The data source and the HetuEngine cluster nodes can communicate with each other.
- If Kerberos authentication is enabled for the cluster (the cluster is in security mode), create a HetuEngine administrator user. If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), create a HetuEngine service user, and assign the HDFS administrator permission to the user. That is, the user is added to both the **hadoop** and **hadoopmanager** user groups. For details about how to create a user, see **Creating a HetuEngine Permission Role**.
- A HetuEngine compute instance has been created. For details, see **Creating a HetuEngine Compute Instance**.
- You have obtained the IP address, port, database instance name or PDB name, username, and password of the Oracle database.

Constraints on the Interconnection with Oracle Data Sources

- Currently, the Oracle data source is read-only by default. Oracle 12 and later versions are supported.
- The schema and table names of Oracle data sources supported by HetuEngine are case insensitive.
- The following syntaxes are not supported: CREATE SCHEMA, ALTER SCHEMA, DROP SCHEMA, INSERT INTO, SELECT, INSERT OVERWRITE, UPDATE, ANALYZE and VIEW.
- Columns of the CLOB, NCLOB, BLOB, or RAW(n) Oracle types or columns of the Trino data type that are mapped to these types do not support predicate pushdown.

  For example, **name** is a column of the VARCHAR type in HetuEngine and is mapped to NCLOB in Oracle. The predicates of the following two queries are not pushed down:

  ```
  SELECT * FROM nation WHERE name>'abcd';
  SELECT * FROM nation WHERE name='abcd';
  ```

- If you specify a WHERE clause, DELETE can be executed only when the predicate in the WHERE clause can be completely pushed down to Oracle.

Configuring the Oracle Data Source

Installing a cluster client

Step 1 Install the cluster client that contains the HetuEngine service in the **/opt/hadoopclient** directory.

Preparing the Oracle driver

Step 2 Obtain the Oracle driver file from the Oracle official website. The format is **ojdbc***xxx***.jar**, for example, **ojdbc8.jar**.

The name of the driver file to be uploaded must meet the following verification rules: The prefix of the driver file must be **ojdbc** and the end be **jar**. The version number in the middle can be any characters, but the total length cannot exceed 80 characters and the file cannot be larger than 100 MB.

Step 3 Upload the Oracle driver file to the cluster where HetuEngine is deployed.

You can use either of the following methods:

- Upload the file to HDFS on FusionInsight Manager.
  a. Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HDFS**.
  b. In the **Basic Information** area on the **Dashboard** page, click the link next to **NameNode Web UI**.
  c. Choose **Utilities** > **Browse the file system** and click  to create the **/user/hetuserver/fiber/extra_file/driver/oracle** directory.
  d. Go to the **/user/hetuserver/fiber/extra_file/driver/oracle** directory and click  to upload the Oracle driver file obtained in **Step 2**.
  e. Click the value in the **Permission** column in the row containing the driver file, select **Read** and **Write** in the **User** column, **Read** in the **Group** column, and **Read** in the **Other** column, and click **Set**.
- Run HDFS commands to upload the file.
  a. Log in to the node where the HDFS service client is deployed and switch to the client installation directory, for example, **/opt/hadoopclient**.
     cd /opt/hadoopclient
  b. Configure environment variables.
     source bigdata_env
  c. If the cluster is in security mode, run the following command to authenticate the user. In normal mode, user authentication is not required.
     kinit *HetuEngine administrator username*
     Enter the password as prompted.
  d. Create the **/user/hetuserver/fiber/extra_file/driver/oracle** directory, upload the Oracle driver obtained in **Step 2**, and modify the permission.
     hdfs dfs -mkdir -p /user/hetuserver/fiber/extra_file/driver/oracle
     hdfs dfs -put ./*Oracle driver file* **/user/hetuserver/fiber/extra_file/driver/oracle**
     hdfs dfs -chmod -R 644 /user/hetuserver/fiber/extra_file/driver/oracle

Configuring the Oracle data source

Step 4 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**.

Step 5 In the **Basic Information** area in the **Dashboard** tab, click the link next to **HSConsole Web UI** to access the HSConsole page.

Step 6 Choose **Data Source** and click **Add Data Source**. Configure parameters on the **Add Data Source** page.

1. Configure the basic information, enter the data source name, and select **JDBC** > **Oracle** as the data source type.
2. Configure the parameters in the **Oracle Configuration** area. For details, see **Table 1 Oracle configuration**.

Table 11-34 Oracle configuration

Parameter	Description	Example Value
Driver Name	Select the Oracle driver that has been uploaded in **Step 2**. The format is *ojdbcxxx*.**jar**.	ojdbc8.jar
JDBC URL	JDBC URL for connecting to the Oracle database. The default port is 1521. The following formats are available: – **jdbc:oracle:thin***@IP address of the Oracle database*:*Port*/*Oracle* PDB name – **jdbc:oracle:thin***@IP address of the Oracle database*:*Port*:*OracleDatabase instance name*	– jdbc:oracle:thin:@192.168.1.1:1521/orclpdb – jdbc:oracle:thin:@192.168.1.1:1521:orcl
Username	Oracle username for connecting to the Oracle data source.	-
Password	Oracle user password for connecting to the Oracle data source.	-

3. (Optional) Customize the configuration.

   Click **Add** to add custom configuration parameters. For details, see **Table 11-35**.

Table 11-35 Custom parameters for the Oracle data source

Parameter	Description	Example Value
case-insensitive-name-matching	The schema and table names of the Oracle data source supported by HetuEngine are case sensitive. – **false** (default value): Schemas and tables whose names contain only lowercase letters can be queried. – **true**: ▪ If no schema or table is matched ignoring case sensitivity, the schema and table can be queried. ▪ If schemas and tables are matched ignoring case sensitivity, the schema and table cannot be queried.	false
case-insensitive-name-matching.cache-ttl	Timeout interval for caching case-sensitive schema and table names of the Oracle data source. The default value is **1m** (1 minute).	1m
dynamic-filtering.enabled	Whether dynamic filters will be pushed down to JDBC queries. – **true** (default value): Enable pushdown. – **false**: Disable pushdown.	true
dynamic-filtering.wait-timeout	Maximum duration for which HetuEngine waits to collect dynamic filters from where the connection is built before starting a JDBC query. Using a larger value may result in a more detailed dynamic filter. However, the latency of some queries is increased. The default value is 20s.	20s
unsupported-type-handling	How data types that are not supported by the connector will be processed. – **CONVERT_TO_VARCHAR**: Convert unsupported types to **VARCHAR** and allow only read operations on them. Only the types in the data type mapping table are supported. – **IGNORE** (default value): Do not display the unsupported types.	IGNORE

Parameter	Description	Example Value
join-pushdown.enabled	Whether to enable join pushdown. Join pushdown may adversely affect some query statements. – **true**: Enable Join pushdown. – false (default value): Disable join pushdown.	false
join-pushdown.strategy	Policy used to evaluate whether Join is pushed down. – **AUTOMATIC** (default value): Enable cost-based connection pushdown. – **EAGER**: Push down joins as much as possible. Even if table statistics are unavailable, using **EAGER** will push down joins, which may cause query performance deterioration. Use **EAGER** only in test and troubleshooting scenarios.	AUTOMATIC
oracle.number.default-scale	Number of decimal places of the HetuEngine Decimal type mapped to the Oracle Number data type (which does not have specified precision and decimal places). By default, this parameter is not set and the column is not supported.	5
oracle.remarks-reporting.enabled	Whether to expose metadata comments. – **true**: Expose metadata annotations. – **false** (default): Metadata annotations are not exposed through the REMARKS column.	false
oracle.synonyms.enabled	Whether to enable the synonym function. HetuEngine disables the support for Oracle SYNONYM by default for performance reasons. – **true**: The synonym function is enabled. – **false** (default value): The synonym function is disabled.	false
oracle.source-encoding	Character set code of the remote data source environment, which is used to prevent garbled characters – ZHS16GBK or GBK – AL32UTF8 or UTF-8 (default value)	UTF-8

You can click **Delete** to delete custom configuration parameters.

4. Click **OK**

Step 7 Log in to the node where the cluster client is deployed and run the following commands to switch to the client installation directory and authenticate the user:

cd /opt/hadoopclient

source bigdata_env

kinit *User performing HetuEngine operations* (If the cluster is in normal mode, skip this command.)

Step 8 Log in to the catalog of the data source.

hetu-cli --catalog *Data source name* **--schema** *Database name*

For example, run the following command:

hetu-cli --catalog oracle_1 --schema oralce

Step 9 Run the following command. If the database table information can be viewed or no error is reported, the connection is successful.

show tables;

----End

Mapping Between Oracle and HetuEngine Data Types

Constraints:

- HetuEngine cannot directly read data of the Number type whose precision and scale are not set in the Oracle database. You need to add the custom parameter **oracle.number.default-scale=s** to the data source configuration to map the data type to the **decimal(38, s)** type.
- HetuEngine cannot read columns whose p – s > 38 in the **Number(p, s)** data type.
- The date type in the Oracle database is stored in seconds. Therefore, the data type mapped to HetuEngine is **timestamp(0)**.
- If the decimal precision of the second in the timestamp data queried by HetuEngine is greater than 3, a value is truncated to three decimal places instead of being rounded off.
- The time and date formats supported by the JDBC driver are different. So, an error may occur when the date and time earlier than **1582-10-15** are inserted or queried.
- **VARCHAR(n)** in HetuEngine is mapped to **VARCHAR2(n CHAR)** in Oracle. If **n** is greater than 4000, it is mapped to **NCLOB**. **CHAR(n)** is mapped to **CHAR(n CHAR)**. If **n** is greater than 2000, it is also mapped to **NCLOB**.
- HetuEngine cannot write **CHAR** and **VARCHAR** data types to columns with improper lengths.
- When you use **CREATE TABLE AS** to create an **NCLOB** column from a **CHAR** value, the suffix space in the initial value is removed. However, if you insert a **CHAR** value into an existing **NCLOB** column, the suffix space is retained.

Table 11-36 Mapping from Oracle to HetuEngine data types

Oracle Type	HetuEngine Data Type
NUMBER(p, s)	DECIMAL(p, s)
NUMBER(p)	DECIMAL(p, 0)
FLOAT[(p)]	DOUBLE
BINARY_FLOAT	REAL
BINARY_DOUBLE	DOUBLE
VARCHAR2(n CHAR)	VARCHAR(n)
VARCHAR2(n BYTE)	VARCHAR(n)
NVARCHAR2(n)	VARCHAR(n)
CHAR(n)	CHAR(n)
NCHAR(n)	CHAR(n)
CLOB	VARCHAR
NCLOB	VARCHAR
RAW(n)	VARBINARY
BLOB	VARBINARY
DATE	TIMESTAMP(0)
TIMESTAMP(p)	TIMESTAMP(3)
TIMESTAMP(p) WITH TIME ZONE	TIMESTAMP(3) WITH TIME ZONE

Table 11-37 Mapping from HetuEngine to Oracle data types

HetuEngine Type	Oracle Type
TINYINT	NUMBER(3)
SMALLINT	NUMBER(5)
INTEGER	NUMBER(10)
BIGINT	NUMBER(19)
DECIMAL(p, s)	NUMBER(p, s)
REAL	BINARY_FLOAT
DOUBLE	BINARY_DOUBLE
VARCHAR	NCLOB
VARCHAR(n)	VARCHAR2(n CHAR) or NCLOB

HetuEngine Type	Oracle Type
CHAR(n)	CHAR(n CHAR) or NCLOB
VARBINARY	BLOB
DATE	DATE
TIMESTAMP	TIMESTAMP(3)
TIMESTAMP WITH TIME ZONE	TIMESTAMP(3) WITH TIME ZONE

Function Enhancement

The connector allows you to directly query the underlying database. Only the native Oracle syntax is supported because complete statements are pushed down to and executed in the Oracle database. The connector can be used to access Oracle functions that are unavailable in HetuEngine or when you want to execute some statements faster on the Oracle side.

```
SELECT * FROM TABLE(
  oracle.system.query(
    query => 'SELECT * FROM tpch.nation'));
```

11.5.11 Adding a GBase Data Source

This topic is available for MRS 3.5.0 and later versions only.

HetuEngine allows you to configure, access, and query the GBase data source. This topic guides you to add an GBase JDBC data source on the HSConsole page of the cluster.

Prerequisites

- The data source and the HetuEngine cluster nodes can communicate with each other.
- In the **/etc/hosts** file of all nodes in the cluster where HetuEngine is deployed, add the host names in the cluster where the data source to be interconnected is deployed and the IP address mappings.
- If Kerberos authentication has been enabled for the cluster (security mode), create a HetuEngine administrator. If Kerberos authentication has been disabled for the cluster (normal mode), create a HetuEngine service user and grant the HDFS administrator permission to the user. That is, when you create a user, add the user to both the **hadoop** and **hadoopmanager** user groups, for details about how to create a user, see **Creating a HetuEngine Permission Role**.
- A HetuEngine compute instance has been created. For details, see **Creating a HetuEngine Compute Instance**.
- You have obtained the IP address, port number, username, and password for logging in to the GBase database.

Constraints on the Interconnection with GBase Data Sources

- HetuEngine supports interconnecting with GBase using the following SQL syntaxes: SHOW CATALOGS, SCHEMAS, TABLES, COLUMNS, DESCRIBE, USE, and SELECT TABLE/VIEW.
- The schema and table names of GBase data sources supported by HetuEngine are case insensitive.

Configuring the GBase Data Source

Installing a cluster client

Step 1 Install the cluster client that contains the HetuEngine service in the **/opt/hadoopclient** directory.

Preparing the GBase driver

Step 2 Obtain the GBase driver file in JAR format from GBase's official website. The version must be **gbase-connector-java-9.5.0.1-build1-bin.jar** or later.

Step 3 Upload the GBase driver file to the cluster where HetuEngine is deployed.

You can use either of the following methods:

- Upload the file to HDFS on FusionInsight Manager.
  a. Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HDFS**.
  b. In the **Basic Information** area on the **Dashboard** page, click the link next to **NameNode Web UI**.
  c. Choose **Utilities** > **Browse the file system** and click  to create the **/user/hetuserver/fiber/extra_file/driver/gbase** directory.
  d. Go to the **/user/hetuserver/fiber/extra_file/driver/gbase** directory and click  to upload the GBase driver file obtained in **Step 2**.
  e. Click the value in the **Permission** column in the row containing the driver file, select **Read** and **Write** in the **User** column, **Read** in the **Group** column, and **Read** in the **Other** column, and click **Set**.
- Run HDFS commands to upload the file.
  a. Upload the obtained GBase driver file to any directory on the node where the HDFS service client is deployed.
  b. Log in to the node where the HDFS service client is deployed and switch to the client installation directory, for example, **/opt/hadoopclient**.
     cd /opt/hadoopclient
  c. Configure environment variables.
     source bigdata_env
  d. If the cluster is in security mode, run the following command to authenticate the user. In normal mode, user authentication is not required.
     kinit *HetuEngine administrator username*
     Enter the password as prompted.

e. Create the **/user/hetuserver/fiber/extra_file/driver/gbase** directory, upload the GBase driver obtained in **Step 2**, and modify the permission.

hdfs dfs -mkdir -p /user/hetuserver/fiber/extra_file/driver/gbase

hdfs dfs -put *GBase driver file path* **/user/hetuserver/fiber/extra_file/driver/gbase**

hdfs dfs -chmod -R 644 /user/hetuserver/fiber/extra_file/driver/gbase

Configuring the GBase data source

Step 4 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**.

Step 5 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**.

Step 6 Choose **Data Source** and click **Add Data Source**. Configure parameters on the **Add Data Source** page.

1. Configure the basic information, enter the data source name, and select **JDBC** > **GBase** as the data source type.
2. In the **GBase Configuration** area, configure the parameters according to **Table 11-38**.

Table 11-38 GBase configurations

Parameter	Description	Example Value
Driver Name	Select the GBase driver that has been uploaded in **Step 2**. The driver format is *xxx*.jar.	gbase-connector-java-9.5.0.1-build1-bin.jar
JDBC URL	JDBC URL for connecting to the GBase database. Format: **jdbc:mysql://***IP address of the GBase database*:*Port number*. The default port is 5258.	jdbc:gbase://192.168.1.1:5258
Username	GBase username for connecting to the GBase data source.	-
Password	GBase password for connecting to the GBase data source.	-

3. (Optional) Customize the configuration.

   Click **Add** to add custom configuration parameters. Configure custom parameters of the GBase data source. For details, see **Table 11-39**.

Table 11-39 Custom parameters for the GBase data source

Parameter	Description	Example Value
GBase.auto-reconnect	Whether to reconnect automatically. – **true** (default value): Enable automatic reconnection. – **false**: Disable automatic reconnection.	true
GBase.max-reconnects	Maximum number of reconnection attempts. The default value is 3.	3
GBase.jdbc.use-information-schema	Whether the driver should use INFORMATION_SCHEMA to derive the information used by DatabaseMetaData.	true
use-connection-pool	Whether to use the JDBC connection pool. The default value is **true**.	true
jdbc.connection.pool.maxTotal	Maximum number of connections in the JDBC connection pool. The default value is **8**.	8
jdbc.connection.pool.maxIdle	Maximum number of idle connections in the JDBC connection pool. The default value is **8**.	8
jdbc.connection.pool.minIdle	Minimum number of idle connections in the JDBC connection pool. The default value is **0**.	0

Parameter	Description	Example Value
unsupported-type-handling	How data types that are not supported by the connector will be processed. – **CONVERT_TO_VARCHAR**: Convert unsupported types to **VARCHAR** and allow only read operations on them. – **IGNORE** (default value): Do not display the unsupported types.	IGNORE
join-pushdown.enabled	Whether join pushdown is enabled. – **true** (default value): Enable join pushdown. – **false**: Disable join pushdown.	true

You can click **Delete** to delete custom configuration parameters.

4. Click **OK**

Step 7 Log in to the node where the cluster client is deployed and run the following commands to switch to the client installation directory and authenticate the user:

cd /opt/hadoopclient

source bigdata_env

kinit *User performing HetuEngine operations* (If the cluster is in normal mode, skip this command.)

Step 8 Log in to the catalog of the data source.

hetu-cli --catalog *Data source name* **--schema** *Database name*

For example, run the following command:

hetu-cli --catalog gbase_1 --schema gbasedb

Step 9 Run the following command. If the database table information can be viewed or no error is reported, the connection is successful.

show tables;

----End

Mapping Between GBase and HetuEngine Data Types

Table 11-40 Mapping Between GBase and HetuEngine Data Types

GBase Type	HetuEngine Type
TINYINT	TINYINT
SMALLINT	SMALLINT
INTEGER	INTEGER
BIGINT	BIGINT
DOUBLE	DOUBLE
FLOAT	REAL
DECIMAL(p, s)	DECIMAL(p, s)
CHAR(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
TEXT	VARCHAR(65535)
BLOB, LONGBLOB	VARBINARY
DATE	DATE
TIME	TIME
DATETIME	TIMESTAMP(6)
TIMESTAMP(n)	TIMESTAMP(n)

11.6 Configuring HetuEngine Materialized Views

11.6.1 Overview of HetuEngine Materialized Views

The materialized views of the HetuEngine are available for MRS 3.2.0 and later versions.

HetuEngine Materialized Views Background

HetuEngine provides the materialized view capability. It enables you to pre-compute frequently accessed and time-consuming operators (such as join and aggregation operators) through materialized views. In this way, queries or subqueries that can match the materialized views are converted into corresponding materialized views, avoiding repeated data computing and improving the query response efficiency.

A materialized view is typically created based on the results of queries that aggregate and join multiple data tables.

Materialized views support query rewrite. It is an optimization technique that converts query statements compiled based on an original table into equivalent requests for querying one or more materialized view statements. The following is an example of the SQL statement of a materialized view:

```
create materialized view mv.default.mv1 with(storage_table='hive.default.mv1') AS select id from hive.mvschema.t1;
```

The actual data of the materialized view is stored in the **hive.default.mv1** table. During query rewriting, the SQL statement **select id from hive.mvschema.t1** is rewritten as the table for querying the materialized view, that is, **select id from hive.default.mv1**.

HetuEngine Materialized Views Scenario

Compared with common views, materialized views occupy storage resources and cause data delay because of actual data storage and pre-computation. Therefore, materialized views are recommended in the following scenarios:

- Frequently executed queries are required.
- Queries involve time-consuming operations like aggregation and join operations.
- A certain delay is allowed for the query result data.
- Materialized views can only be connected to co-deployed Hive and external Hive data sources. Data source tables are stored in ORC or PARQUET format. Cross-source and cross-domain scenarios are not supported.

HetuEngine Materialized Views Permission Introduction

Table 11-41 lists materialized view permissions. Permission control for materialized views depends on the Ranger. If Ranger authentication is disabled, permissions may become invalid.

Table 11-41 HetuEngine materialized view permissions

Operation	Permission on catalog mv	Permission on Tables Stored in MVs	Permission on Original Physical Table
Creating a materialized view	Permission to create tables	NA	Column query permission
Deleting a materialized view	Permission to delete tables	N/A	N/A
Refreshing a materialized view	Permission to update tables	N/A	Column query permission
Modifying the properties or state of a materialized view	Permission to alter tables	NA	NA

Operation	Permission on catalog mv	Permission on Tables Stored in MVs	Permission on Original Physical Table
Overwriting query statements using materialized views	N/A	N/A	Column query permission
Using materialized views to rewrite the execution plan of query statements (EXPLAIN)	N/A	Column query permission	Column query permission
Querying a materialized view	Column query permission	N/A	N/A
Querying physical tables of materialized and non-materialized views	Column query permission	N/A	Column query permission
Viewing a materialized view	N/A	N/A	N/A
Viewing the statement for creating a materialized view	Permission to show tables	Permission to show tables	N/A

How to Use HetuEngine Materialized Views

Table 11-42 Introduction to materialized views

Phase	Description	Reference
SQL statement example of materialized views	This section describes the operations supported by materialized views, including creating, listing, and querying materialized views.	**SQL Examples of HetuEngine Materialized Views**
Configuring rewriting of materialized views	Enables the materialized view capability for faster query response.	**Rewriting of HetuEngine Materialized Views**
Configuring recommendation of materialized views	Automatically learns and recommends materialized view SQL statements that are most valuable to services, improving online query efficiency and reducing system load pressure.	**HetuEngine Materialized View Recommendation**

Phase	Description	Reference
Configuring caching of materialized views	The SQL statements that have been executed and rewritten for multiple times can be saved to the cache. When the SQL statements are executed again, the rewritten SQL statements are directly obtained from the cache instead of rewriting the SQL statements, improving query efficiency.	**HetuEngine Materialized View Caching**
Configuring the validity period and data update of materialized views	• Configures the validity period of the materialized view. Currently, only the materialized view within the validity period is automatically overwritten. • Configures periodic data update. Materialized views can be refreshed manually or automatically.	**Validity Period and Data Update of HetuEngine Materialized Views**
Configuring intelligent materialized views	Provides automatic creation of materialized views. You do not need to manually execute SQL statements to create materialized views (recommended).	**HetuEngine Intelligent Materialized Views**
Viewing automatic tasks of materialized views	Views the task execution status to evaluate the cluster health status.	**Automatic Tasks of HetuEngine Materialized Views**

11.6.2 SQL Examples of HetuEngine Materialized Views

For details about the SQL statements for materialized views, see **Table 11-43**.

Table 11-43 Operations on materialized views

Operation	Function	SQL Statement Example of Materialized View	Remarks
Creating a materialized view (When a materialized view is created, only the definition of the materialized view is created. To fill in data, run the **refresh materialized view name** command.)	Create a materialized view that never expires.	create materialized view mv.default.mv1 with(storage_table='hive.default.mv11') AS select id from hive.mvschema.t1;	• **storage_table** specifies the location where the materialized view data is materialized into a physical table. • When creating a materialized view, you must specify **mv** for the catalog. You can also create a schema. • For the **AS SELECT** clause, pay attention to the items listed in **Creating the AS SELECT Clause for a Materialized View**.
	Create a materialized view that is valid for one day and cannot automatically refresh.	create materialized view mv.default.mv1 with(storage_table='hive.default.mv11', mv_validity = '24h') AS select id from hive.mvschema.t1;	**mv_validity** specifies the validity of a materialized view.

Operation	Function	SQL Statement Example of Materialized View	Remarks
	Create a materialized view that automatically refreshes data every hour.	create materialized view mv.default.mv1 with(storage_table='hive.default.mv1', need_auto_refresh = true, mv_validity = '1h', start_refresh_ahead_of_expiry = 0.2, refresh_priority = 3, refresh_duration = '5m') AS select id from hive.mvschema.t1;	• **need_auto_refresh**: indicates whether to enable automatic refresh. • **start_refresh_ahead_of_expiry**: a refresh task is triggered for the materialized view at the time specified by **mv_validity*** (**1-start_refresh_ahead_of_expiry**) so that the task status is changed to **Refreshable**. • **refresh_priority** specifies the priority of refreshing tasks. • **refresh_duration** specifies the maximum duration of a refreshing task.
Showing materialized views	Show all MVs whose catalog name is **mv** and schema name is **mvschema**.	show materialized views from mvschema;	**mvschema** indicates the schema name. The value of **catalog** is fixed to **mv**.
	Use the LIKE clause to filter the materialized views whose names meet the rule expression.	show MATERIALIZED VIEWs in mvschema tables like '*mvtb_0001';	**mvschema** indicates the schema name.

Operation	Function	SQL Statement Example of Materialized View	Remarks
Querying the statement for creating a materialized view	Query the statement for creating the the materialized view of **mv.default.mv1**.	show create materialized view mv.default.mv1;	**mv1** indicates the name of the materialized view.
Querying a materialized view	Query data in **mv.default.mv1**.	select * from mv.default.mv1;	**mv1** indicates the name of the materialized view.
Refreshing a materialized view	Refresh the materialized view of **mv.default.mv1**.	refresh materialized view mv.default.mv1;	-
Modifying the properties of materialized views	Modifying the properties of the **mv.default.mv1** materialized view	Alter materialized view mv.mvtestprop.pepa_ss set PROPERTIES refresh_priority = 2;	**refresh_priority = 2** is the property of the materialized view.

Operation	Function	SQL Statement Example of Materialized View	Remarks
Changing the status of materialized views	Changing the status of the **mv.default.mv1** materialized view	alter materialized view mv.default.mv1 set status SUSPEND;	**SUSPEND** is the status of the materialized view. The status can be: • **SUSPEND**: The materialized view is suspended. The suspended materialized view is not rewritten. • **ENABLE**: The materialized view is available. • **REFRESHING**: The materialized view data is being refreshed and cannot be rewritten. • **DISABLE**: The materialized view is disabled. You can only convert the status between **ENABLE** and **SUSPEND**, and chang the **DISABLE** state to **SUSPEND** or **ENABLE**.
Deleting a materialized view	Delete the materialized view of **mv.default.mv1**.	drop materialized view mv.default.mv1;	-
Enabling materialized view rewriting capability to optimize SQL statements	Enabling materialized view rewriting capability at the session level to optimize SQL statements	set session materialized_view_rewrite_enabled=true;	-

Operation	Function	SQL Statement Example of Materialized View	Remarks
Verifying whether SQL statements can be optimized by rewriting a query to a materialized view	Verify whether the SELECT statement can be rewritten and optimized by **mv.default.mv1**.	verify materialized view mvname(mv.default.mv1) originalsql select id from hive.mvschema.t1;	-
Enabling the specified materialized view at the SQL level to optimize the SQL statements	Forcibly use **mv.default.mv1** for SQL statement optimization in queries.	/*+ REWRITE(mv.default.mv1) */ select id from hive.mvschema.t1;	-
Disabling materialized views at the SQL level to optimize the SQL statements	Do not use materialized views for SQL statement optimization in queries.	/*+ NOREWRITE */ select id from hive.mvschema.t1;	-
Refreshing the metadata cache of materialized views	Synchronize the metadata cache of materialized views between tenants.	refresh catalog mv;	-

Creating the AS SELECT Clause for a Materialized View

The **AS SELECT** clause for creating materialized views cannot contain reserved keywords in Calcite SQL parsing and rewriting functions, such as **default**. To use reserved keywords in the **AS SELECT** clause, use either of the following solutions:

- When creating MVs and executing original queries, you need to add double quotes to the default schema name.

  The following uses reserved keyword **default** in the **AS SELECT** clause as an example:

  Creating a materialized view

  ```
  CREATE MATERIALIZED VIEW mv.default.mv1 WITH(storage_table='hive.default.mv11') AS SELECT id FROM hive."default".t1;
  ```

  SELECT query

  ```
  SELECT id FROM hive."default".t1;
  ```

- Set the corresponding catalog and schema at the Session level, rather than passing fully qualified names in the query.

  For example, set **catalogname** to **hive** and **schemaname** to **default**.

  ```
  USE hive.default;
  ```

  Creating a materialized view

  ```
  CREATE MATERIALIZED VIEW mv.default.mv1 WITH(storage_table='hive.default.mv11') AS SELECT id FROM t1;
  ```

  SELECT query

  ```
  SELECT id FROM t1;
  ```

11.6.3 Rewriting of HetuEngine Materialized Views

Enabling Rewriting of Materialized Views

HetuEngine provides the materialized view rewriting capability at the system or session level.

- Enabling the materialized view rewriting capability at the session level:

  Run the **set session materialized_view_rewrite_enabled=true** command on the HetuEngine client

- Enabling the materialized view rewriting capability at the system level:

  a. Log in to FusionInsight Manager as a user who can access the HetuEngine web UI.

  b. Choose **Cluster** > **Services** > **HetuEngine** to go its service page.

  c. In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

  d. Click **Compute Instance** to view the instance status of the tenant to which operations are to be performed. When the number of green and blue icons is 0, you can perform **e** to enable materialized view rewriting.

  e. In the **Compute Instance** page, locate the row that contains the tenant to which the target instance belongs and click **Configure** in the **Operation** column. On the tab page displayed, add the following custom parameters:

Table 11-44 Custom parameters

Parameter	Value	Parameter File
materialized.view.rewrite.enabled	true	coordinator.config.properties
materialized.view.rewrite.timeout	5	coordinator.config.properties

NOTE

This step applies to MRS 3.2.0 or later.

- **materialized.view.rewrite.timeout**: timeout interval for overwriting a materialized view, in seconds. The recommended value is 5 seconds. Materialized view rewrite takes some time. This parameter can be added to limit the performance loss caused by rewrite. After materialized view rewrite times out, the original SQL statement is executed.
- To enable the materialized view function at the session level and enable the timeout control for materialized view rewrite, run the **set session materialized_view_rewrite_timeout = 5** command first.

f. Set **Start Now** to **Yes** and click **OK**.

Scope of Materialized View Rewriting

- Supported materialized view types

  BOOLEAN, DECIMAL, DOUBLE, REAL/FLOAT, INT, BIGINT, SMALLINT, TINYINT, CHAR/VARCHAR, DATE, TIME, TIMESTAMP, INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND, BINARY/VARBINARY, and UUID.

- Supported functions for materialized view rewriting
  - Conversion function: Only the CAST function is supported.
  - String function: All string functions are supported, including char_length, character_length, chr, codepoint, decode, encode, find_in_set, format_number, locate, hamming_distance, instr, levenshtein, levenshtein_distance, ltrim, lpad, octet_length, position, quote, and repeat2.
  - Mathematical operator: All mathematical operators are supported.
  - Aggregate function: **COUNT**, **SUM**, **MIN**, **MAX**, **AVG**, **LEAD**, **LAG**, **FIRST_VALUE**, **LAST_VALUE**, **COVAR_POP**, **COVAR_SAMP**, **REGR_SXX**, **REGR_SYY**, **STDDEV_POP**, **STDDEV_SAMP**, **VAR_POP**, **VAR_SAMP**, **ROW_NUMBER**, **RANK**, **PERCENT_RANK**, **DENSE_RANK**, and **CUME_DIST** are supported.

NOTICE

In the following scenarios, materialized views cannot be used to rewrite SQL queries that contain functions:

- SQL queries contain parameterless functions.
- SQL queries contain functions supported by HetuEngine that obtain different types of return values based on parameter types.
- SQL queries contain nested functions or functions that may throw exceptions causing rewrite failures.

- The statement for creating materialized views does not support the table name that contains two elements.

  For example, the table names **hive.mvschema.t1** and **t1** are supported, while the table name **mvschema.t1** cannot be used.

Example of Materialized View Rewriting Scenarios

The core principle of materialized view rewriting is that the data of the logically created materialized view must contain the data to be queried in the future query statements or all the data to be included in the subquery in the future query. It is recommended that you enable the automatic creation of materialized views to create materialized views. The following is an example of some scenarios:

In the SQL statement example for creating a materialized view, **CREATE MATERIALIZED VIEW** *xxx* **WITH(***xxx***) AS** is omitted. For details about the complete statement template, see **Table 11-43**.

Table 11-45 Example of materialized view rewriting scenarios

Scenario	Description	SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
Full table query	Basic full table query scenario	select * from tb_a;	select * from tb_a;	No	Creating a materialized view for full table scanning is meaningless and is not supported.
Column query	Basic column query scenario	select col1,col2,col3 from tb_a;	select col1,col2,col3 from tb_a;	Yes	-

Scenario	Description	SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
	User query renaming	select col1 from tb_a;	select col1 as a from tb_a;	Yes	-
		select col1,col2,col3 from tb_a;	select col1 as a,col2 as b,col3 as c from tb_a;	Yes	-
	Mathematical expression	select col1*col2 from tb_a;	select col2*col1 from tb_a;	Yes	The two columns must have the same type.

Scenario	Description	SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
	Source column used by a materialized view; and **cast** is used for user query.	select col1,col2 from tb_a;	select cast(col1 as varchar),col2 from tb_a;	No	Original data columns used by a materialized view, which are not rewritten if no filter criteria are configured in the functions used for user query. Original data columns used by a materialized view, which can be rewritten if the original data columns and filter criteria are used for user query.
	case when scenario	select col1, (case col2 when 1 then 'b' when 2 'a' end) as col from tb_a;	select col1, (case col2 when 1 then 'b' when 2 'a' end) as col from tb_a;	No	The case when scenario is not supported in query columns.

Scenario	Description		SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
	String function		select col13 from tb_a;	select length(col13) from tb_a;	No	All string functions use the original table data to create materialized views. The materialized views are not rewritten when queries without filter criteria configured.
			select length(col13) from tb_a;	select length(col13) from tb_a;	Yes	-
Aggregate function column query	count	Materialized views and user queries use **count**.	select count(col1) from tb_a;	select count(col1) from tb_a;	Yes	-

Scenario	Description		SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
		Source data used by a materialized view, and **count** is used for user queries.	select col1 from tb_a;	select count(col1) from tb_a;	Yes	-
	sum	**sum** is used for materialized views and user queries.	select sum(col1) from tb_a;	select sum(col1) from tb_a;	Yes	-
		Source data used by a materialized view, and **sum** is used for user queries.	select col1 from tb_a;	select sum(col1) from tb_a;	Yes	-

Scenario	Description		SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
Querying information by specifying filter criteria (The core is that the data in materialized views is logically the same as or more than that in query SQL statements.)	where filtering	Maximum range of materialized views (<)	select col1 from tb_a;	select col1 from tb_a where col1<11;	Yes	-
		The materialized view range is greater than the user query range (<).	select col1 from tb_a where col1<50;	select col1 from tb_a where col1<45;	Yes	-
			select col1 from tb_a where col1<50;	select col1 from tb_a where col1<=45;	Yes	-
			select col1 from tb_a where col1<50;	select col1 from tb_a where col1 between 21 and 29;	Yes	-
		The materialized view range is equal to the user query range (>).	select col1 from tb_a where col1<50;	select col1 from tb_a where col1<50;	Yes	-
		The materialized view range is greater than the user query range (and).	select col1 from tb_a where col1<60 and col1>30;	select col1 from tb_a where col1<55 and col1>30;	Yes	-
			select col1 from tb_a where col1<60 and col1>30;	select col1 from tb_a where col1 between 35 and 55;	Yes	-

Scenario	Description		SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
			select col1 from tb_a where col1<60 and col1>30;	select col1 from tb_a where (col1<55 and col1>30) and col1 = 56;	Yes	-
	where nested subquery	Subquery source table as a materialized view	select col1 from tb_a;	select count(col1) from tb_a where col1=(select min(col1) from tb_a);	Yes	-
		Subquery as a materialized view	select min(col1) from tb_a;	select count(col1) from tb_a where col1=(select min(col1) from tb_a);	Yes	-
		Parent query source table as a materialized view	select col1 from tb_a where col1=(select min(col1) from tb_a);	select count(col1) from tb_a where col1=(select min(col1) from tb_a);	Yes	-
		Parent query as a materialized view	select count(col1) from tb_a where col1=(select min(col1) from tb_a);	select count(col1) from tb_a where col1=(select min(col1) from tb_a);	Yes	-
	limit	limit in a query	select col1 from tb_a;	select col1 from tb_a limit 5;	Yes	-

Scenario	Description		SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
			select col1 from tb_a limit 5;	select col1 from tb_a limit 5;	Yes	-
			select col1 from tb_a limit 5;	select col1 from tb_a;	No	-
		limit combined with order by	select col1 from tb_a;	select col1 from tb_a order by col1 limit 5;	Yes	If **order by** is used to create a materialized view, the result may be disordered. If query rewrite for materialized views is enabled, do not use **limit** or **order by** in the materialized view creation statement.
			select col1 from tb_a order by col1;	select col1 from tb_a order by col1 limit 5;	Yes	
			select col1 from tb_a order by col1 limit 5;	select col1 from tb_a order by col1 limit 5;	No	

Scenario	Description		SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
	having filtering	Maximum range of materialized views (<)	select col1 from tb_a;	select col1 from tb_a group by col1 having col1 <11;	Yes	group by + having: The scenario of having is different from that of where. The having condition cannot be compensated. The materialized view SQL statements must not have the having condition or must be the same as that of user query SQL statements.
		The materialized view range is greater than the user query range (<).	select col1 from tb_a group by col1 having col1<50;	select col1 from tb_a group by col1 having col1<45;	No	
			select col1 from tb_a group by col1 having col1<50;	select col1 from tb_a group by col1 having col1<=45;	No	
			select col1 from tb_a group by col1 having col1<50;	select col1 from tb_a group by col1 having col1=45;	No	
			select col1 from tb_a group by col1 having col1<50;	select col1 from tb_a group by col1 having col1 between 21 and 29;	No	
		The materialized view range is greater than the user query range (<).	select col1 from tb_a group by col1 having col1<50;	select col1 from tb_a group by col1 having col1<50;	Yes	

Scenario	Description	SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
JOIN association query	Two subqueries as a materialized view	select col1,col3 from tb_a where col1<11;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select col1,col2 from t1 join t2 on t1.col3=t2.col3;	Yes	-
		select cast(col2 as varchar) col2,col3 from tb_b;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select col1,col2 from t1 join t2 on t1.col3=t2.col3;	Yes	-
	Parent query as a materialized view	with t1 as (select col1,col3 from tb_a),t2 as (select col2,col3 from tb_b) select col1,col2 from t1 join t2 on t1.col3=t2.col3;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select col1,col2 from t1 join t2 on t1.col3=t2.col3;	Yes	-

Scenario	Description	SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
Aggregate + JOIN query	Source table data as a materialized view	select col1,col3 from tb_a;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	Yes	-
		select col2,col3 from tb_b;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	Yes	-
	Subquery as a materialized view	select col1,col3 from tb_a where col1<11;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	Yes	-

Scenario	Description	SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
		select cast(col2 as varchar) col2,col3 from tb_b;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	Yes	-
	Parent query (whose subqueries use the source table, non-aggregate query) as a materialized view	with t1 as (select col1,col3 from tb_a),t2 as (select col2,col3 from tb_b) select col1,col2 from t1 join t2 on t1.col3=t2.col3;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	Yes	-

Scenario	Description	SQL Statement Example for Creating a Materialized View	SQL Statement Example for a User Query	SQL Statement Rewritable	Remarks
	Parent query (non-aggregate query) as a materialized view	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select col1,col2 from t1 join t2 on t1.col3=t2.col3;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	Yes	-
	Parent query as a materialized view	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	Yes	-

11.6.4 HetuEngine Materialized View Recommendation

HetuEngine QAS module provides automatic detection, learning, and diagnosis of historical SQL execution records. After the materialized view recommendation function is enabled, the system can automatically learn and recommend the most valuable materialized view SQL statements, enabling HetuEngine to have the automatic precomputation acceleration capability. In related scenarios, the online

query efficiency is improved by multiple times, and the system load pressure is effectively reduced.

Prerequisites

- The cluster is running properly and at least one QAS instance has been installed.
- You have created a user for accessing the HetuEngine web UI, for example, **Hetu_user**. For details, see **Creating a HetuEngine Permission Role**.

Enabling Materialized View Recommendation

Step 1 Log in to FusionInsight Manager as user **Hetu_user**.

Step 2 Choose **Cluster** > **Services** > **HetuEngine** and then choose **Configurations** > **All Configurations**. In the navigation tree, choose **QAS(Role)** > **Materialized View Recommendation**. Set materialized view recommendation parameters by referring to **Table 11-46** and retain the default values for other parameters.

Table 11-46 Materialized view recommendation parameters

Parameter	Example Value	Description
qas.enable.auto.recommendation	true	Whether to enable materialized view recommendation. The default value is **false**.
qas.sql.submitter	default,zuhu1	Name of the tenant for which the materialized view recommendation function is enabled. Use commas (,) to separate multiple tenants.
qas.schedule.fixed.delay	1d	Interval for recommending materialized views. Once a day is recommended.
qas.threshold.for.mv.recommend	0.05	Filtering threshold of materialized view recommendation. The value ranges from **0.001** to **1**. You are advised to adjust the value based on the site requirements.

Step 3 Click **Save**.

Step 4 Click **Instance**, select all QAS instances, click **More**, and select **Restart Instance**. In the displayed dialog box, enter the password to restart all QAS instances for the parameters to take effect.

----End

Viewing Materialized View Recommendation Results

Step 1 Log in to FusionInsight Manager as user **Hetu_user**.

Step 2 Choose **Cluster** > **Services** > **HetuEngine** to go its service page.

Step 3 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 4 Choose **SQL O&M** > **Automatic MV Recommendation**. You can search for materialized views by tenant, status, recommendation period, and materialized view name. Fuzzy search is supported. You can export the recommendation result of a specified materialized view.

The status of a materialized view task can be:

Table 11-47 Status of a materialized view task

Status Name	Description	Status Name	Description
To Be Created	To be created	Deleting	Terminating
Creating	Creating	Deleted	Terminated
Created	Created	Planning	Being planned
Failed	Creation failed	Aborted	Aborted
Updating	Updating	Duplicated	Repeated recommendation

----End

11.6.5 HetuEngine Materialized View Caching

After a materialized view is created for an SQL statement, the SQL statement is rewritten to be queried through the materialized view when the SQL statement is executed. If the rewrite cache function is enabled for materialized views, the rewritten SQL statements will be saved to the cache (a maximum of 10,000 records can be saved by default) after the SQL statement is executed for multiple times. When the SQL statement is executed within the cache validity period (24 hours by default), the system obtains the rewritten SQL statement from the cache instead of rewriting the SQL statement.

You can add user-defined parameters **rewrite.cache.timeout** and **rewrite.cache.limit** to a compute instance to set the cache validity period and the maximum number of rewritten SQL statements that can be saved.

- When a new materialized view is created or an existing materialized view is deleted, the cache becomes invalid.
- If the materialized view associated with a rewritten SQL query in the cache becomes invalid or is in the **Refreshing** status, the rewritten SQL query will not be used.
- When the cache is used, the executed SQL query cannot be changed. Otherwise, it will be treated as a new SQL query.

- A maximum of 500 materialized views can be rewritten for SQL queries. That is, if the materialized views used during SQL rewriting are included in the 500 materialized views, the views will be rewritten. Otherwise, the views will be executed as common SQL statements. You can refer to **System level** to add user-defined parameter **hetu.select.top.materialized.view** to compute instances to change the number of materialized views that can be used.

Enabling Rewrite Cache for Materialized Views

- Session level:

  Run the **set session rewrite_cache_enabled=true** command on the HetuEngine client.

- Enabling the materialized view rewriting capability at the system level:

  a. Log in to FusionInsight Manager as a user who can access the HetuEngine web UI.

  b. Choose **Cluster** > **Services** > **HetuEngine** to go its service page.

  c. In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

  d. Click **Compute Instance** to view the instance status of the tenant to which operations are to be performed. When the number of green and blue icons is 0, you can perform **e** to enable materialized view rewriting.

  e. In the **Compute Instance** page, locate the row that contains the tenant to which the target instance belongs and click **Configure** in the **Operation** column. On the tab page displayed, add the following custom parameters:

Parameter	Value	Parameter File	Description
rewrite.cache.enabled	true	coordinator.config.properties	Enable the rewrite cache function for materialized views.
rewrite.cache.timeout	86400000	coordinator.config.properties	• Change the validity period of the rewrite cache. • If this parameter is left blank, **86400000** is used by default. The unit is ms.
rewrite.cache.limit	10000	coordinator.config.properties	• Modify the upper limit of the rewrite cache. • If this parameter is left blank, **10000** is used by default.

  f. Set **Start Now** to **Yes** and click **OK**.

11.6.6 Validity Period and Data Update of HetuEngine Materialized Views

Validity Period of Materialized Views

The **mv_validity** field for creating a materialized view indicates the validity period of the materialized view. HetuEngine allows you to rewrite the SQL statements using only the materialized views within the validity period.

Refreshing Materialized View Data

If data needs to be updated periodically, you can use either of the following methods to periodically refresh the materialized views:

- Manually refreshing a materialized view

  Run the **refresh materialized view** *<mv name>* command on the HetuEngine client, or run the **refresh materialized view** *<mv name>* command using JDBC in the service program to manually update the database.

- Automatically refreshing a materialized view

  a. To enable the automatic refresh capability of the materialized views, you must set a compute instance as the maintenance instance. For details, see **Configuring a HetuEngine Maintenance Instance**.

  b. Use **create materialized view** to create a materialized view that can be automatically refreshed.

  NOTE

  - If there are too many materialized views, some materialized views may expire due to too long waiting time.
  - The automatic refresh function does not automatically refresh materialized views in the **disable** status.

Automatically Refreshing Materialized Views When Querying External Hive Data Sources

By default, the maintenance instance uses the HetuEngine built-in user **hetuserver/hadoop.<*System domain name*>** for refreshing materialized views. When materialized view creation statements query external Hive data sources where authentication has been enabled, you need to change the user for automatically refreshing materialized views as follows:

Step 1 Check whether the HetuEngine service has been installed in the peer cluster.

- If yes, go to **Step 3**.
- If no, go to **Step 2**.

Step 2 Prepare the user used by the system for automated refresh.

1. Create a human-machine user with the same name in both the local and peer clusters.

   Take **mvuser** as an example. In the peer cluster, add it to the **supergroup** user group. In the local cluster, add it to the **supergroup** and **hive** user groups and add the tenant role of the maintenance instance.

2. (Optional) If Ranger authentication is enabled in the local cluster, grant the permission to refresh materialized views and **set sessions** permission to the **mvuser** user. For details, see **Table 11-41** and **Table 23-21**.

Step 3 Log in to FusionInsight Manager as the HetuEngine administrator.

Step 4 Choose **Cluster** > **Services** > **HetuEngine** and click the **Configurations** tab and then **All Configurations**.

Step 5 Search for **jobsystem.customized.properties**, add a custom configuration named **hetuserver.engine.jobsystem.inner.principal**, and set its value according to the following content. Then, click **Save** and operate as prompted.

- If the HetuEngine service has been installed in the peer cluster, set the value to **hetuserver**.
- If the HetuEngine service is not installed in the peer cluster, set the value the user name created in **Step 2.1**.

Step 6 Click the **Instance** tab, select all HSBroker instances, click **More**, and select **Restart Instance** to restart the HSBroker instances as prompted.

Step 7 In the displayed **Dashboard** tab, find the **Basic Information** area, and click the link next to **HSConsole WebUI**. In the **Compute Instance** tab, locate the maintenance instance and click **Restart** in the **Operation** column and operate as prompted.

NOTE

Instances with the  icon displayed next to their names are maintenance instances. You can also confirm the maintenance instance by referring to **Configuring a HetuEngine Maintenance Instance**.

----End

11.6.7 HetuEngine Intelligent Materialized Views

Overview

HetuEngine intelligent materialized views provide intelligent precalculation and cache acceleration. The HetuEngine QAS role can automatically extract historical SQL statements for analysis and learning, and automatically generate candidate SQL statements for high-value materialized views based on the revenue maximization principle. In practice, HetuEngine administrators can enable automatic creation and refresh of materialized views by configuring maintenance instances. Service users can configure client sessions to implement automatic rewriting and acceleration based on automatically created materialized views.

This capability significantly simplifies the use of materialized views and accelerates analysis without interrupting services. HetuEngine administrators can intelligently accelerate high-frequency SQL services by using a small amount of compute and storage resources. In addition, this capability reduces the overall load (such as CPU, memory, and I/O) of the data platform and improves system stability.

The intelligent materialized view provides the following functions:

- Automatic recommendation of materialized views

- Automatic creation of materialized views
- Automatic refresh of materialized views
- Automatic deletion of materialized views

Prerequisites

The cluster is running properly and at least one QAS instance has been installed.

Application Process

Figure 11-4 Application process of HetuEngine intelligent materialized views

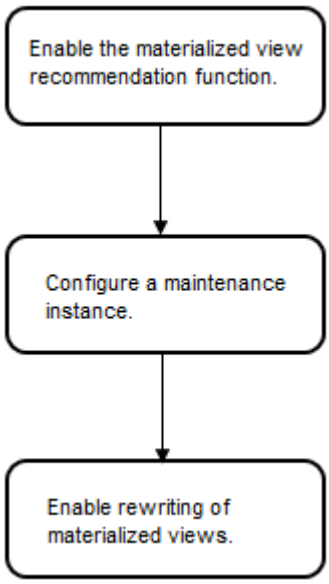


Table 11-48 Process description

Step	Description	Reference
Enable the materialized view recommendation function.	After this function is enabled, QAS instances automatically recommend SQL statements of high-value materialized views based on users' SQL execution records. You can view the recommended materialized view statements on the materialized view recommendation page on the HSConsole. For details, see **Viewing Materialized View Recommendation Results**.	**Enabling Materialized View Recommendation**

Step	Description	Reference
Configure a maintenance instance.	After a compute instance is set as a maintenance instance, the maintenance instance automatically creates, refreshes, and deletes the materialized view SQL statements recommended by the materialized view recommendation function. You can view the generated automatic task records on the HetuEngine automation task page. For details, see **Automatic Tasks of HetuEngine Materialized Views**.	**Configuring a HetuEngine Maintenance Instance**
Enable rewriting of materialized views.	After rewriting is enabled for materialized views, HetuEngine determines whether the materialized view rewriting requirements are met based on the SQL statements entered by users and converts queries or subqueries that match materialized views into materialized views, avoiding repeated data calculation.	**Rewriting of HetuEngine Materialized Views**

11.6.8 Automatic Tasks of HetuEngine Materialized Views

View the status and execution result of an automatic HetuEngine task on HSConsol. You can periodically view the task execution status and evaluate the cluster health status.

Procedure for Automatic Tasks of HetuEngine Materialized Views

Step 1 Log in to FusionInsight Manager as a user who can access the HetuEngine web UI and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 2 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 3 Choose **Automated Tasks**. On the displayed page, you can search for tasks by **Task Type**, **Status**, **Additional Info**, or **Start Time**. Fuzzy search is supported.

Search Criterion	Description
Task Type	**ALL**: all types
	Refresh of materialized view: refreshes materialized views.
	Recommendation of materialized view: recommends materialized views.

Search Criterion	Description
	Auto create materialized view: automatically creates materialized views.
	Drop auto created and stale materialized view: automatically deletes materialized views.
Status	ALL
	success
	failed
	waiting
	running
	skipped
	time out
	unknown

Step 4 Click **Query**. The tasks that match the search criteria are displayed.

----End

11.7 HetuEngine SQL Diagnosis

Scenario

The HetuEngine QAS module provides automatic detection, learning, and diagnosis of historical SQL execution records for more efficient online SQL O&M and faster online SQL analysis. After SQL diagnosis is enabled, the system provides the following capabilities:

- Automatically detects and displays tenant-level and user-level SQL execution statistics in different time periods to cluster administrators, helping them quickly predict service running status and potential risks.
- Automatically diagnoses large SQL statements, slow SQL statements, and related submission information, displays the information in multiple dimensions for cluster administrators, and provides diagnosis and optimization suggestions for these statements.

This section applies to MRS 3.2.0 or later.

Prerequisites

- The cluster is running properly and at least one QAS instance has been installed.
- You have created a user for accessing the HetuEngine web UI, for example, **Hetu_user**. For details, see **Creating a HetuEngine Permission Role**.

Enabling SQL Diagnosis

The SQL diagnosis function of HetuEngine is enabled by default. You can perform the following steps to configure other common parameters or retain the default settings:

Step 1 Log in to FusionInsight Manager as user **Hetu_user**.

Step 2 Choose **Cluster** > **Services** > **HetuEngine**. Click **Configurations** then **All Configurations**, click **QAS(Role)**, and select **SQL Diagnosis**. If **qas.sql.auto.diagnosis.enabled** is set to **true**, the SQL diagnosis function is enabled. In this case, you can configure recommended SQL diagnosis parameters based on service requirements.

Step 3 Click **Save**.

Step 4 Click **Instance**, select all QAS instances, click **More**, and select **Restart Instance**. In the displayed dialog box, enter the password to restart all QAS instances for the parameters to take effect.

----End

Viewing SQL Diagnosis Results

Step 1 Log in to FusionInsight Manager as user **Hetu_user**.

Step 2 Choose **Cluster** > **Services** > **HetuEngine** to go its service page.

Step 3 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 4 Choose **SQL O&M** to view SQL diagnosis results.

- On the **Overview** page, you can view the overall running status of historical tasks, including the query duration distribution chart by segment, query user distribution chart, total submitted SQL queries, SQL execution success rate, average SQL query response time, number of queries, average execution time, and average waiting time.
- On the **Slow Query Distribution** page, view the slow query distribution of historical tasks, including:
  - Slow SQL statistics: collects statistics on the number of slow queries (the query time is greater than the slow query threshold) submitted by each tenant.
  - Top users with the maximum slow query requests: collects statistics on slow query statistics of each user. The statistics can be sorted in a list and exported.
- On the **Slow Queries** page, view the slow query list of historical tasks, diagnosis results, and optimization suggestions. Query results can be exported.

  NOTE

  The validity period of historical statistics depends on the JVM memory size of HSConsole instances and cannot exceed 60 days.

----End

11.8 Developing and Deploying HetuEngine UDFs

11.8.1 Developing and Deploying HetuEngine Function Plugins

You can customize functions to extend SQL statements to meet personalized requirements. These functions are called UDFs.

This section describes how to develop and apply HetuEngine function plugins.

NOTE

The development must be based on JDK 17.0.4 or later for MRS 3.2.1 or later. This topic uses MRS 3.3.0 as an example.

Developing Function Plugins

This sample implements two function plugins described in the following table.

Table 11-49 HetuEngine function plugins

Parameter	Description	Type
add_two	Adds **2** to the input integer and returns the result.	ScalarFunction
avg_double	Aggregates and calculates the average value of a specified column. The field type of the column is **double**.	AggregationFunction

Step 1 Create a Maven project. Set **groupId** to **com.test.functions** and **artifactId** to **myfunctions**. The two values can be customized based on the site requirements.

Step 2 Modify the **pom.xml** file as follows:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.test.functions</groupId>
    <artifactId>myfunctions</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>trino-plugin</packaging>
  <properties>
     <project.build.targetJdk>17</project.build.targetJdk>
     <dep.guava.version>31.1-jre</dep.guava.version>
      <dep.hetu.version>399-h0.cbu.mrs.321.r13</dep.hetu.version>
  </properties>

   <dependencies>
      <dependency>
         <groupId>com.google.guava</groupId>
         <artifactId>guava</artifactId>
         <version>${dep.guava.version}</version>
```

```
        </dependency>

        <dependency>
            <groupId>io.trino</groupId>
            <artifactId>trino-spi</artifactId>
            <version>${dep.hetu.version}</version>
            <scope>provided</scope>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-assembly-plugin</artifactId>
                <version>3.3.0</version>
                <configuration>
                    <encoding>UTF-8</encoding>
                </configuration>
            </plugin>
            <plugin>
                <groupId>io.trino</groupId>
                <artifactId>trino-maven-plugin</artifactId>
                <version>11</version>
                <extensions>true</extensions>
            </plugin>
        </plugins>
    </build>
</project>
```

Step 3 Create the implementation class of the function plugin.

1. Create the function plugin implementation class **com.test.functions.scalar.MyFunction**. The content is as follows:

```
package com.test.functions.scalar;
import io.trino.spi.function.ScalarFunction;
import io.trino.spi.function.SqlNullable;
import io.trino.spi.function.SqlType;
import io.trino.spi.type.StandardTypes;
import jdk.jfr.Description;
public class MyFunction {
    private MyFunction() {
    }
    @Description("Add two")
    @ScalarFunction("add_two")
    @SqlType(StandardTypes.INTEGER)
    public static long add2(@SqlNullable @SqlType(StandardTypes.INTEGER) Long i) {
        return i + 2;
    }
}
```

2. Create the function plugin implementation class **com.test.function.aggregation.MyAverageAggregationFunction**. The content is as follows:

```
package com.test.functions.aggregation;

import static io.trino.spi.type.DoubleType.DOUBLE;

import io.trino.spi.block.BlockBuilder;
import io.trino.spi.function.AggregationFunction;
import io.trino.spi.function.AggregationState;
import io.trino.spi.function.CombineFunction;
import io.trino.spi.function.InputFunction;
import io.trino.spi.function.OutputFunction;
import io.trino.spi.function.SqlType;
import io.trino.spi.type.StandardTypes;

@AggregationFunction("avg_double")
```



```
public class MyAverageAggregationFunction
{
    private MyAverageAggregationFunction() {}

    @InputFunction
    public static void input(
        @AggregationState LongAndDoubleState state,
        @SqlType(StandardTypes.DOUBLE) double value)
    {
        state.setLong(state.getLong() + 1);
        state.setDouble(state.getDouble() + value);
    }

    @CombineFunction
    public static void combine(
        @AggregationState LongAndDoubleState state,
        @AggregationState LongAndDoubleState otherState)
    {
        state.setLong(state.getLong() + otherState.getLong());
        state.setDouble(state.getDouble() + otherState.getDouble());
    }

    @OutputFunction(StandardTypes.DOUBLE)
    public static void output(@AggregationState LongAndDoubleState state, BlockBuilder out)
    {
        long count = state.getLong();
        if (count == 0) {
            out.appendNull();
        }
        else {
            double value = state.getDouble();
            DOUBLE.writeDouble(out, value / count);
        }
    }
}
```

Step 4 Create the **com.test.functions.aggregation.LongAndDoubleState** API on which AverageAggregation depends.

```
package com.test.functions.aggregation;

import io.trino.spi.function.AccumulatorState;

public interface LongAndDoubleState
    extends AccumulatorState
{
    long getLong();

    void setLong(long value);

    double getDouble();

    void setDouble(double value);
}
```

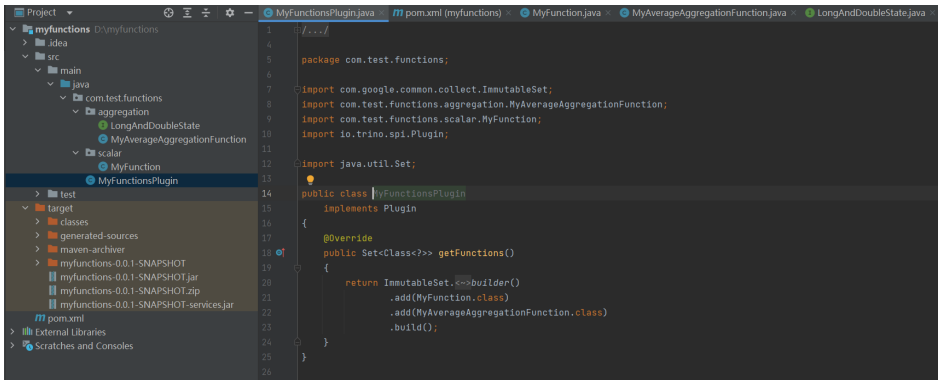
Step 5 Create the function plugin registration class **com.test.functions.MyFunctionsPlugin**. The content is as follows:

```
package com.test.functions;

import com.google.common.collect.ImmutableSet;
import com.test.functions.aggregation.MyAverageAggregationFunction;
import com.test.functions.scalar.MyFunction;

import io.trino.spi.Plugin;

import java.util.Set;

public class MyFunctionsPlugin
    implements Plugin
```



```
{
    @Override
    public Set<Class<?>> getFunctions() {
        return ImmutableSet.<Class<?>>builder()
            .add(MyFunction.class)
            .add(MyAverageAggregationFunction.class)
            .build();
    }
}
```

Step 6 Pack the Maven project and obtain the **myfunctions-0.0.1-SNAPSHOT** directory in the **target** directory. The following figure shows the overall structure of the project:



----End

Deploying Function Plugins

Before the deployment, ensure that:

- The HetuEngine service is normal.
- The HDFS and HetuEngine client have been installed in a directory on the cluster node, for example, **/opt/client**.
- A HetuEngine user has been created. For details about how to create a user, see **Creating a HetuEngine Permission Role**.

Step 1 Upload the **myfunctions-0.0.1-SNAPSHOT** directory obtained in packing the Maven project to any directory on the node where the client is installed.

Step 2 Upload the **myfunctions-0.0.1-SNAPSHOT** directory to HDFS.

NOTE

You need to upload all JAR packages in the target directory in **Step 6**.

1. Log in to the node where the client is installed and perform security authentication.

   cd /opt/client

   source bigdata_env

   kinit *HetuEngine user*

   Enter the password as prompted and change the password upon the first authentication.

2. Create the following paths in HDFS. If the paths already exist, skip this step.

   hdfs dfs -mkdir -p /user/hetuserver/udf/data/externalFunctionsPlugin

3. Upload the **myfunctions-0.0.1-SNAPSHOT** directory to HDFS.

   hdfs dfs -put myfunctions-0.0.1-SNAPSHOT /user/hetuserver/udf/data/externalFunctionsPlugin

4. Change the directory owner and owner group.

   hdfs dfs -chown -R hetuserver:hadoop /user/hetuserver/udf/data

Step 3 Restart the HetuEngine compute instance.

----End

Verifying Function Plugins

Step 1 Log in to the node where the client is installed and perform security authentication.

cd /opt/client

source bigdata_env

kinit *HetuEngine user*

hetu-cli

Step 2 Select columns that have numeric values (int or double type) in a table from the verification environment. In this example, table **hive.default.test1** is used. Run the following command to verify the function plugins:

1. Query a table.

   select * from hive.default.test1;

```
select * from hive.default.test1;
name  |  price
--------|-------
apple   |  17.8
orange |  25.0
(2 rows)
```

2. Return the average value.

   select avg_double(price) from hive.default.test1;

```
select avg_double(price) from hive.default.test1;
_col0
-------
 21.4
(1 row)
```

3. Return the value of the input integer plus 2.

   select add_two(4);

```
select add_two(4);
_col0
-------
   6
(1 row)
```

----End

11.8.2 Hive UDFs for Interconnecting with HetuEngine

You can customize functions to extend SQL statements to meet personalized requirements. These functions are called UDFs.

This section describes how to develop and apply Hive UDFs.

NOTE

The development must be based on JDK 17.0.4 or later for MRS 3.2.1 or later. This topic uses MRS 3.3.0 as an example.

Developing Hive UDFs

This sample implements one Hive UDF described in the following table.

Table 11-50 Hive UDF

Parameter	Description
AutoAddOne	Adds **1** to the input value and returns the result.

NOTE

- A common Hive UDF must be inherited from **org.apache.hadoop.hive.ql.exec.UDF**.
- A common Hive UDF must implement at least one **evaluate()**. The **evaluate** function supports overloading.
- Currently, only the following data types are supported:
  - boolean, byte, short, int, long, float, and double
  - Boolean, Byte, Short, Int, Long, Float, and Double
  - List and Map

  UDFs, UDAFs, and UDTFs currently do not support complex data types other than the preceding ones.
- Currently, Hive UDFs supports only less than or equal to five input parameters. UDFs with more than five input parameters will fail to be registered.
- If the input parameter of a Hive UDF is **null**, the call returns **null** directly without parsing the Hive UDF logic. As a result, the UDF execution result may be inconsistent with the Hive execution result.
- To add the **hive-exec-3.1.1** dependency package to the Maven project, you can obtain the package from the Hive installation directory.
- (Optional) If the Hive UDF depends on a configuration file, you are advised to save the configuration file as a resource file in the **resources** directory so that it can be packed into the Hive UDF function package.

Step 1 Create a Maven project. Set **groupId** to **com.test.udf** and **artifactId** to **udf-test**. The two values can be customized based on the site requirements.

Step 2 Modify the **pom.xml** file as follows:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.test.udf</groupId>
    <artifactId>udf-test</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencies>
      <dependency>
        <groupId>org.apache.hive</groupId>
        <artifactId>hive-exec</artifactId>
        <version>3.1.1</version>
```

```
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <artifactId>maven-shade-plugin</artifactId>
        <executions>
          <execution>
            <phase>package</phase>
            <goals>
              <goal>shade</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
      <plugin>
        <artifactId>maven-resources-plugin</artifactId>
        <executions>
          <execution>
            <id>copy-resources</id>
            <phase>package</phase>
            <goals>
              <goal>copy-resources</goal>
            </goals>
            <configuration>
              <outputDirectory>${project.build.directory}/</outputDirectory>
              <resources>
                <resource>
                  <directory>src/main/resources/</directory>
                  <filtering>false</filtering>
                </resource>
              </resources>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

Step 3 Create the implementation class of the Hive UDF.

```
import org.apache.hadoop.hive.ql.exec.UDF;

/**
 * AutoAddOne
 *
 * @since 2020-08-24
 */
public class AutoAddOne extends UDF {
    public int evaluate(int data) {
        return data + 1;
    }
}
```

Step 4 Package the Maven project. The **udf-test-0.0.1-SNAPSHOT.jar** file in the **target** directory is the Hive UDF function package.

NOTE

You need to pack all dependencies into a JAR package.

----End

Configuring Hive UDFs

In configuration file **udf.properties**, add registration information in the "Function_name Class_path" format to each line.

The following provides an example of registering four Hive UDFs in configuration file **udf.properties**:

```
booleanudf io.hetu.core.hive.dynamicfunctions.examples.udf.BooleanUDF
shortudf io.hetu.core.hive.dynamicfunctions.examples.udf.ShortUDF
byteudf io.hetu.core.hive.dynamicfunctions.examples.udf.ByteUDF
intudf io.hetu.core.hive.dynamicfunctions.examples.udf.IntUDF
```

NOTE

- If the added Hive UDF registration information is incorrect, for example, the format is incorrect or the class path does not exist, the system ignores the incorrect registration information and prints the corresponding logs.
- If duplicate Hive UDFs are registered, the system will only register once and ignore the duplicate registrations.
- If the Hive UDF to be registered is the same as that already registered in the system, the system throws an exception and cannot be started properly. To solve this problem, you need to delete the Hive UDF registration information.

Deploying Hive UDFs

To use an existing Hive UDF in HetuEngine, you need to upload the UDF function package, **udf.properties** file, and configuration file on which the UDF depends to the specified HDFS directory, for example, **/user/hetuserver/udf/**, and restart the HetuEngine compute instance.

Step 1 Create the **/user/hetuserver/udf/data/externalFunctions** directory, save the **udf.properties** file in the **/user/hetuserver/udf** directory, save the UDF function package in the **/user/hetuserver/udf/data/externalFunctions** directory, and save the configuration files on which the UDF depends in the **/user/hetuserver/udf/data** directory.

- Upload the files on the HDFS page:
  a. Log in to FusionInsight Manager using the HetuEngine username and choose **Cluster** > **Services** > **HDFS**.
  b. In the **Basic Information** area on the **Dashboard** page, click the link next to **NameNode WebUI**.
  c. Choose **Utilities** > **Browse the file system** and click  to create the **/user/hetuserver/udf/data/externalFunctions** directory.
  d. Go to **/user/hetuserver/udf** and click  to upload the **udf.properties** file.
  e. Go to the **/user/hetuserver/udf/data/** directory and click  to upload the configuration file on which the UDF depends.
  f. Go to the **/user/hetuserver/udf/data/externalFunctions** directory and click  to upload the UDF function package.
- Use the HDFS CLI to upload the files.
  a. Log in to the node where the HDFS service client is located and switch to the client installation directory, for example, **/opt/client**.
  cd /opt/client

b. Run the following command to configure environment variables:

source bigdata_env

c. If the cluster is in security mode, run the following command to authenticate the user. In normal mode, skip user authentication.

kinit *HetuEngineUsername*

Enter the password as prompted.

d. Run the following commands to create directories and upload the prepared UDF function package, **udf.properties** file, and configuration file on which the UDF depends to the target directories:

hdfs dfs -mkdir /user/hetuserver/udf/data/externalFunctions

hdfs dfs -put ./*Configuration files on which the UDF depends* **/user/hetuserver/udf/data**

hdfs dfs -put ./udf.properties /user/hetuserver/udf

hdfs dfs -put ./*UDF function package* **/user/hetuserver/udf/data/externalFunctions**

Step 2 Restart the HetuEngine compute instance.

----End

Using Hive UDFs

Use a client to access a Hive UDF:

1. Log in to the HetuEngine client.
2. Run the following command to use a Hive UDF:

select AutoAddOne(1);

```
select AutoAddOne(1);
_col0
-------
    2
(1 row)
```

11.8.3 Developing and Deploying HetuEngine UDFs

You can customize functions to extend SQL statements to meet personalized requirements. These functions are called UDFs.

This section describes how to develop and apply HetuEngine UDFs.

NOTE

The development must be based on JDK 17.0.4 or later for MRS 3.2.1 or later. This topic uses MRS 3.3.0 as an example.

Developing HetuEngine UDFs

This sample implements one HetuEngine UDF described in the following table.

Table 11-51 HetuEngine UDF

Parameter	Description
AddTwo	Adds **2** to the input value and returns the result.

Step 1 Create a Maven project. Set **groupId** to **com.test.udf** and **artifactId** to **udf-test**. The two values can be customized based on the site requirements.

Step 2 Modify the **pom.xml** file as follows:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.test.udf</groupId>
    <artifactId>udf-test</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <build>
      <plugins>
        <plugin>
          <artifactId>maven-shade-plugin</artifactId>
          <executions>
            <execution>
              <phase>package</phase>
              <goals>
                <goal>shade</goal>
              </goals>
            </execution>
          </executions>
        </plugin>
        <plugin>
          <artifactId>maven-resources-plugin</artifactId>
          <executions>
            <execution>
              <id>copy-resources</id>
              <phase>package</phase>
              <goals>
                <goal>copy-resources</goal>
              </goals>
              <configuration>
                <outputDirectory>${project.build.directory}/</outputDirectory>
                <resources>
                  <resource>
                    <directory>src/main/resources/</directory>
                    <filtering>false</filtering>
                  </resource>
                </resources>
              </configuration>
            </execution>
          </executions>
        </plugin>
      </plugins>
    </build>
</project>
```

Step 3 Create the implementation class of the HetuEngine UDF.

```
package com.xxxbigdata..hetuengine.functions;

public class AddTwo {
   public Integer evaluate(Integer num) {
      return num + 2;
   }
}
```


Step 4 Package the Maven project. The **udf-test-0.0.1-SNAPSHOT.jar** file in the **target** directory is the HetuEngine UDF function package.

NOTE

- A common HetuEngine UDF must implement at least one **evaluate()**. The **evaluate** function supports overloading.
- Currently, HetuEngine UDFs supports only less than or equal to five input parameters. HetuEngine UDFs with more than five input parameters will fail to be registered.
- You need to pack all dependencies into a JAR package.
- (Optional) If the HetuEngine UDF depends on a configuration file, you are advised to save the configuration file as a resource file in the **resources** directory so that it can be packed into the HetuEngine UDF function package.

----End

Deploying HetuEngine UDFs

To use the HetuEngine UDF in HetuEngine, you need to upload the corresponding UDF function package to a specified HDFS directory, for example, **/udf/hetuserver**. The directory can be customized based on the site requirements.

Create the **/udf/hetuserver** directory and save the UDF function package to it.

- Upload the files on the HDFS page:
  a. Log in to FusionInsight Manager using the HetuEngine username and choose **Cluster** > **Services** > **HDFS**.
  b. In the **Basic Information** area on the **Dashboard** page, click the link next to **NameNode WebUI**.
  c. Choose **Utilities** > **Browse the file system** and click  to create the **/udf/hetuserver** directory.
  d. Go to the **/udf/hetuserver** directory and click  to upload UDF function package.
- Use the HDFS CLI to upload the files.
  a. Log in to the node where the HDFS service client is located and switch to the client installation directory, for example, **/opt/client**.

     cd /opt/client
  b. Run the following command to configure environment variables:

     source bigdata_env
  c. If the cluster is in security mode, run the following command to authenticate the user. In normal mode, skip user authentication.

     kinit*HetuEngineusername*

     Enter the password as prompted.
  d. Run the following commands to create a directory and upload the prepared UDF function package to the target directory:

     hdfs dfs -mkdir -p /udf/hetuserver

     hdfs dfs -put ./*UDF function package* **/udf/hetuserver**.

e. Run the following command to change the permission of the UDF function package:

hdfs dfs -chmod 644 /udf/hetuserver/*UDF function package*

NOTICE

- When uploading a UDF JAR file to a user-defined HDFS directory, ensure that the user has the read permission on the JAR file. You are advised to use **chmod 644** to set the permission. In addition, if you want the UDF JAR file to be deleted during the HetuEngine service uninstallation, you can create a user-defined directory in the **/user/hetuserver/** directory.
- Currently, HetuEngine supports the UDF JAR file to be stored only in **hdfs://** *Resource URI* in HDFS.
- If the JAR file is re-uploaded due to function modification or addition, HetuEngine caches the classloader for 5 minutes by default. The JAR file does not take effect immediately, instead, it is updated and reloaded 5 minutes later.

Using HetuEngine UDFs

Use a client to access a HetuEngine UDF.

1. Log in to the HetuEngine client.
2. Create a HetuEngine UDF.

```
CREATE FUNCTION example.namespace01.add_two (
 num integer
)
RETURNS integer
LANGUAGE JAVA
DETERMINISTIC
SYMBOL "com.xxx.bigdata.hetuengine.functions.AddTwo"
URI "hdfs://hacluster/udf/hetuserver/udf-test-0.0.1-SNAPSHOT.jar";
```

3. Use the HetuEngine UDF.

```
select example.namespace01.add_two(2);
_col0
-------
    4
(1 row)
```

NOTE

- Overloading is used to distinguish functions with the same name in the implementation classes. Therefore, you need to specify different function names when creating a HetuEngine UDF.

11.9 Managing a HetuEngine Data Source

On the HetuEngine web UI, you can view, edit, and delete an added data source.

Step 1 Log in to Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**. The HetuEngine service page is displayed.

Step 2 In the **Basic Information** area on the **Dashboard** tab page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 3 Click **Data Source**. In the data source list, view the data source name, description, type, and creation time. You can also edit or delete a data source in the **Operation** column.

NOTE

During HetuEngine installation, the co-deployed Hive data source is interconnected by default. The data source name is **hive** and cannot be deleted.

----End

11.10 Managing HetuEngine Compute Instances

11.10.1 Configuring HetuEngine Resource Groups

Resource Group Introduction

The resource group mechanism controls the overall query load of the instance from the perspective of resource allocation and implements queuing policies for queries. Multiple resource groups can be created under a compute instance resource, and each submitted query is assigned to a specific resource group for execution. Before a resource group executes a new query, it checks whether the resource load of the current resource group exceeds the amount of resources allocated to it by the instance. If it is exceeded, new incoming queries are blocked, placed in a queue, or even rejected directly.

Application Scenarios of Resource Groups

Resource groups are used to manage resources in compute instances. Different resource groups are allocated to different users and queries to isolate resources. This prevents a single user or query from exclusively occupying resources in the compute instance. In addition, the weight and priority of resource components can be configured to ensure that important tasks are executed first. **Table 11-52** describes the typical application scenarios of resource groups.

Table 11-52 Typical application scenarios of resource groups

Typical Scenarios	Solution
As the number of business teams using the compute instance increases, there is no resource when a team's task becomes more important and does not want to execute a query.	Allocate a specified resource group to each team. Important tasks are assigned to resource groups with more resources. When the sum of the proportions of sub-resource groups is less than or equal to 100%, the resources of a queue cannot be preempted by other resource groups. This is similar to static resource allocation.

Typical Scenarios	Solution
When the instance resource load is high, two users submit a query at the same time. At the beginning, both queries are queuing. When there are idle resources, the query of a specific user can be scheduled to obtain resources first.	Two users are allocated with different resource groups. Important tasks can be allocated to resource groups with higher weights or priorities. The scheduling policy is configured by schedulingPolicy. Different scheduling policies have different resource allocation sequences.
For ad hoc queries and batch queries, resources can be allocated more properly based on different SQL types.	You can match different resource groups for different query types, such as EXPLAIN, INSERT, SELECT, and DATA_DEFINITION, and allocate different resources to execute the query.

Enabling a Resource Group

When creating a compute instance, add custom configuration parameters to the **resource-groups.json** file. For details, see **Step 3.5** in **Creating a HetuEngine Compute Instance**.

Resource Group Properties

For details about how to configure resource group attributes, see **Table 11-53**.

Table 11-53 Resource group properties

Configuration Item	Mandatory	Description
name	Yes	Resource group name
maxQueued	Yes	Maximum number of queued queries. When this threshold is reached, new queries will be rejected.
hardConcurrencyLimit	Yes	Maximum number of running queries.
softMemoryLimit	No	Maximum memory usage of a resource group. When the memory usage reaches this threshold, new tasks are queued. The value can be a fixed value (for example, 10 GB) or a percentage (for example, 10% of the cluster memory).

Configuration Item	Mandatory	Description
softCpuLimit	No	The CPU time that can be used in a period (see the **cpuQuotaPeriod** parameter in **Global Attributes**). You must also specify the **hardCpuLimit** parameter. When the threshold is reached, the CPU resources occupied by the query that occupies the maximum CPU resources in the resource group are reduced.
hardCpuLimit	No	Maximum CPU time that can be used in a period.
schedulingPolicy	No	The scheduling policy for a specific query from the queuing state to the running state • fair (default) When multiple sub-resource groups in a resource group have queuing queries, the sub-resource groups obtain resources in turn based on the defined sequence. The query of the same sub-resource group obtains resources based on the first-come-first-executed rule. • weighted_fair The **schedulingWeight** attribute is configured for each resource group that uses this policy. Each sub-resource group calculates a ratio: *Number of queried sub-resource groups*/Scheduling weight. A sub-resource group with a smaller ratio obtains resources first. • weighted The default value is **1**. A larger value of **schedulingWeight** indicates that resources are obtained earlier. • query_priority All sub-resource groups must be set with **query_priority**. Resources are obtained in the sequence specified by **query_priority**.
schedulingWeight	No	Weight of the group. For details, see **schedulingPolicy**. The default value is **1**.
jmxExport	No	If this parameter is set to **true**, group statistics are exported to the JMX for monitoring. The default value is **false**.
subGroups	No	Subgroup list

Configuration Item	Mandatory	Description
killPolicy	No	After a query is submitted to worker, if the total memory usage exceeds **softMemoryLimit**, you can select one of the following policies to terminate running queries: • **no_kill** (default value): Do not terminate the queries. • **recent_queries**: Terminate the queries based on the execution sequence in descending order. • **oldest_queries**: Terminate the queries based on the execution sequence. • **finish_percentage_queries**: Terminate the queries based on query execution percentage. The query with the smallest percentage of execution will be terminated first. **high_memory_queries**: Terminate the queries based on memory usage. Queries with high memory usage are terminated first to free up more memory with the minimum number of query terminations. If the memory usage of two queries is less than 10%, the query with slower progress (smaller execution percentage) is terminated. If the difference between the execution percentages of two queries is less than 5%, the query with larger memory usage is terminated.

Selector Rules

The selector matches resource groups in sequence. The first matched resource group is used. Generally, you are advised to configure a default resource group. If no default resource group is configured and other resource group selector conditions are not met, the query will be rejected. For details about how to set selector rule parameters, see **Table 11-54**.

Table 11-54 Selector rules

Configuration Item	Mandatory	Description
user	No	Regular expression for matching the user name.
source	No	Request source of the match. For details, see the value of **--source** in **Configuration of Selector Attributes**.

Configuration Item	Mandatory	Description
queryType	No	Task types: • **DATA_DEFINITION**: indicates that you can modify, create, or delete the metadata of schemas, tables, and views, and manage the query of prepared statements, permissions, sessions, and transactions. • **DELETE**: indicates the DELETE queries. • **DESCRIBE**: indicates the DESCRIBE, DESCRIBE INPUT, DESCRIBE OUTPUT, and SHOW queries. • **EXPLAIN**: indicates the EXPLAIN queries. • **INSERT**: indicates the INSERT and CREATE TABLE AS queries. • **SELECT**: indicates the SELECT queries.
clientTags	No	Match client tag to be matched with. Each tag must be in the tag list of the task submitted by the user. For details, see the value of **--client-tags** in **Configuration of Selector Attributes**.
group	Yes	The resource group with running queries

Global Attributes

For details about how to configure global attributes, see **Table 11-55**.

Table 11-55 Global attributes

Configuration Item	Mandatory	Description
cpuQuotaPeriod	No	Time range during which the CPU quota takes effect. This parameter is used together with **softCpuLimit** and **hardCpuLimit** in **Resource Group Properties**.

Configuration of Selector Attributes

The data source name (**source**) can be set as follows:

- **CLI**: Use the **--source** option.
- **JDBC**: Set the ApplicationName client information property on the Connection instance.

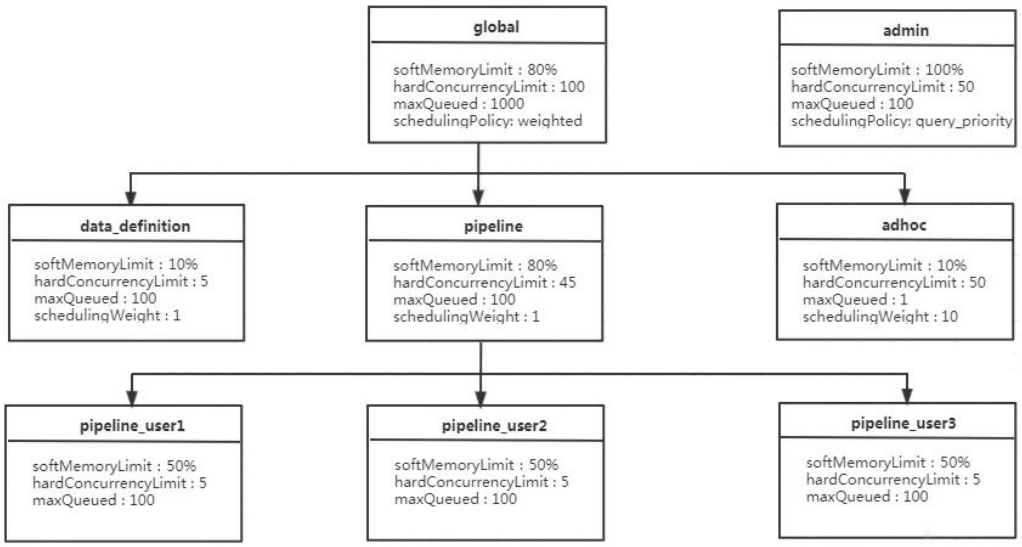
The client tag (**clientTags**) can be configured as follows:

- **CLI**: Use the **--client-tags** option.

- **JDBC**: Set the **ClientTags client info** property on the Connection instance.

Configuration Example

Figure 11-5 Configuration example



As shown in **Figure 11-5**.

- For the **global** resource group, a maximum of 100 queries can be executed at the same time. 1000 queries are in the queuing state. The **global** resource group has three sub-resource groups: **data_definition**, **adhoc**, and **pipeline**.
- Each user in the **pipeline** resource group can run a maximum of five queries at the same time, which occupy 50% of the memory resources of the pipeline resource group. By default, the **fair** scheduling policy is used in the **pipeline** resource group. Therefore, the query is executed in the sequence of "first come, first served".
- To make full use of instance resources, the total memory quota of all child resource groups can be greater than that of the parent resource group. For example, the sum of the memory quota of the **global** resource group (80%) and that of the **admin** resource group (100%) is 180%, which is greater than 100%.

In the following example configuration, there are multiple resource groups, some of which are templates. HetuEngine administrators can use templates to dynamically build a resource group tree. For example, in the **pipeline**_*${USER}* group, *${USER}* is the name of the user who submits a query. *${SOURCE}* is also supported, which will be the source where a query is submitted later. You can also use custom variables in **source** expressions and **user** regular expressions.

The following is an example of a resource group selector:

```
"selectors": [{
   "user": "bob",
   "group": "admin"
},
{
   "source": ".*pipeline.*",
   "queryType": "DATA_DEFINITION",
```



```
    "group": "global.data_definition"
},
{
    "source": ".*pipeline.*",
    "group": "global.pipeline.pipeline_${USER}"
},
{
    "source": "jdbc#(?<toolname>.*)",
    "clientTags": ["hipri"],
    "group": "global.adhoc.bi-${toolname}.${USER}"
},
{
    "group": "global.adhoc.other.${USER}"
}]
```

There are four selectors that define which resource group to run the query:

- The first selector matches queries from **bob** and places them in the **admin** group.
- The second selector matches all data definition language (DDL) queries from the source name that includes the **pipeline** and places them in the **global.data_definition** group. This helps reduce the queuing time of such queries.
- The third selector matches queries from sources that include the **pipeline** and places them in a single-user pipe group that is dynamically created under the **global.pipeline** group.
- The fourth selector matches queries from BI tools whose source matches the regular expression **jdbc#(?.*)**, and the tags provided by the client are the superset of **hi-pri**. These queries are placed in subgroups dynamically created under the **global.adhoc** group. Dynamic subgroups are created based on the naming variable **toolname** that is extracted from the regular expression of the source. Assume that there is a query whose source is **jdbc#powerfulbi**, user is **kayla**, and client labels are **hipri** and **fast**. This query will be routed to the **global.adhoc.bi-powerfulbi.kayla** resource group.
- The last selector is a default selector that puts all the unmatched queries into the resource group.

These selectors work together to implement the following policies:

- HetuEngine administrator **bob** can run 50 queries concurrently. Queries are run in a sequence of priority in descending order.
- For the remaining users:
  - The total number of concurrent queries cannot exceed 100.
  - You can use the source pipeline to run a maximum of five concurrent DDL queries. The query is performed in the FIFO sequence.
  - Non-DDL queries are executed in the **global.pipeline** group. The total number of concurrent queries is 45, and each user can run 5 queries concurrently. The query is performed in the FIFO sequence.
  - Each BI tool can run a maximum of 10 concurrent queries, and each user can run a maximum of three concurrent queries. If the total number of concurrent queries exceeds 10, the user who runs the least queries gets the next concurrency slot. This policy makes it fairer to compete for resources.
  - All the remaining queries are placed in each of the user groups under **global.adhoc.other**.

The description of the query match selector is as follows:

- Each pair of braces represents a selector that matches the resource group. Five selectors are configured to match the five resource groups.

```
admin
global.data_definition
global.pipeline.pipeline_${USER}
global.adhoc.bi-${toolname}.${USER}
global.adhoc.other.${USER}
```

- Only when all the conditions of the selector are met, the task can be put into the current queue for execution. For example, if user **amy** submits a query request in JDBC mode and **clientTags** is not configured, the query request cannot be allocated to the resource group **global.adhoc.bi-${toolname}.${USER}**.
- When a query meets the conditions of two selectors at the same time, the first selector that meets the requirements is matched. For example, if the **bob** user submits a DATA_DEFINITION job whose source is **pipeline**, only the resource corresponding to the resource group **admin** is matched, not the resource corresponding to **global.data_definition**.
- If none of the four selectors is matched, resources in the resource group **global.adhoc.other.${USER}** specified by the last selector are used. This resource group functions as a default resource group. If the default resource group is not set and does not meet the conditions of other resource group selectors, the resource group will be rejected.

  The following is a complete example:

```
{
  "rootGroups": [{
    "name": "global",
    "softMemoryLimit": "80%",
    "hardConcurrencyLimit": 100,
    "maxQueued": 1000,
    "schedulingPolicy": "weighted",
    "jmxExport": true,
    "subGroups": [{
      "name": "data_definition",
      "softMemoryLimit": "10%",
      "hardConcurrencyLimit": 5,
      "maxQueued": 100,
      "schedulingWeight": 1
    },
    {
      "name": "adhoc",
      "softMemoryLimit": "10%",
      "hardConcurrencyLimit": 50,
      "maxQueued": 1,
      "schedulingWeight": 10,
      "subGroups": [{
        "name": "other",
        "softMemoryLimit": "10%",
        "hardConcurrencyLimit": 2,
        "maxQueued": 1,
        "schedulingWeight": 10,
        "schedulingPolicy": "weighted_fair",
        "subGroups": [{
          "name": "${USER}",
          "softMemoryLimit": "10%",
          "hardConcurrencyLimit": 1,
          "maxQueued": 100
        }]
      },
      {
        "name": "bi-${toolname}",
```



```
            "softMemoryLimit": "10%",
            "hardConcurrencyLimit": 10,
            "maxQueued": 100,
            "schedulingWeight": 10,
            "schedulingPolicy": "weighted_fair",
            "subGroups": [{
               "name": "${USER}",
               "softMemoryLimit": "10%",
               "hardConcurrencyLimit": 3,
               "maxQueued": 10
            }]
         }]
      },
      {
         "name": "pipeline",
         "softMemoryLimit": "80%",
         "hardConcurrencyLimit": 45,
         "maxQueued": 100,
         "schedulingWeight": 1,
         "jmxExport": true,
         "subGroups": [{
            "name": "pipeline_${USER}",
            "softMemoryLimit": "50%",
            "hardConcurrencyLimit": 5,
            "maxQueued": 100
         }]
      }]
   },
   {
      "name": "admin",
      "softMemoryLimit": "100%",
      "hardConcurrencyLimit": 50,
      "maxQueued": 100,
      "schedulingPolicy": "query_priority",
      "jmxExport": true
   }],
   "selectors": [{
      "user": "bob",
      "group": "admin"
   },
   {
      "source": ".*pipeline.*",
      "queryType": "DATA_DEFINITION",
      "group": "global.data_definition"
   },
   {
      "source": ".*pipeline.*",
      "group": "global.pipeline.pipeline_${USER}"
   },
   {
      "source": "jdbc#(?<toolname>.*)",
      "clientTags": ["hipri"],
      "group": "global.adhoc.bi-${toolname}.${USER}"
   },
   {
      "group": "global.adhoc.other.${USER}"
   }],
   "cpuQuotaPeriod": "1h"
}
```

11.10.2 Configuring the Number of HetuEngine Worker Nodes

Scenario for Configuring the Number of HetuEngine Worker Nodes

On the HetuEngine web UI, you can adjust the number of worker nodes for a compute instance so that worker nodes can be added when they are insufficient

and reduced when they are idle. The number of worker nodes can be adjusted manually or automatically.

- When an instance is being scaled in or out, the original services are not affected and the instance can still be used.
- Dynamic instance scale-in/out is delayed to implement smooth adjustment of resource consumption within a long period of time. It cannot respond to the requirements of running SQL tasks for available resources in real time.
- After instances are dynamically scaled in or out, the number of worker nodes displayed in the instance configuration area on the HSConsole page remains the initial value and does not change with dynamic scaling.
- The dynamic instance scale-in/out function will be affected if the HSBroker and Yarn services are restarted after the function is enabled. Disable the function before you restart the services.
- Before scaling out a compute instance, ensure that the current queue has sufficient resources. Otherwise, the scale-out cannot reach the expected result and subsequent scale-in operations will be affected.
- You can set the timeout period for manual instance scale in/out. For this, log in to Manager, choose **HetuEngine** > **Configurations** > **All Configurations**, search for **application.customized.properties**, and add the **yarn.hetuserver.engine.flex.timeout.sec** parameter. The default value is **300** (in seconds).
- You can set whether to wait for tasks when YARN resources are insufficient during manual scale-out.

  On FusionInsight Manager, choose **HetuEngine** > **Configurations** > **All Configurations**, search for **application.customized.properties**, and add the **yarn.hetuserver.engine.worker.scale.out.resource.limit** parameter, the options are as follows:

  - **true** (default value): The system checks the YARN resources usage. If the resources are sufficient, they are used for scale-out. If the resources are insufficient, they cannot be used for scale-out.
  - **false**: Scale-out tasks are delivered to YARN without calculating whether there are sufficient Yarn resources. If the resources are sufficient, they are used for scale-out. If the resources are insufficient, the scale-out task waits in a queue.

Procedure for Configuring the Number of HetuEngine Worker Nodes

Step 1 Log in to FusionInsight Manager as a user who can access the HetuEngine web UI and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 2 In the **Basic Information** area on the **Dashboard** tab page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 3 In the **Compute Instance** page, locate the row that contains the tenant to which the target instance belongs and click **Configure** in the **Operation** column.

- If manual scaling is required, change the value of **Quantity** in the **Worker Container Resource Configuration** area on the configuration page and click **OK**. The compute instance status changes to **SCALING OUT** or **SCALING IN**. After the scaling is complete, the compute instance status changes to **RUNNING**.

- If automatic scaling is required, set **Scaling** in **Advanced Configuration** to **Yes** and configure the following parameters according to **Table 11-56** to enable dynamic scaling.

  NOTE

  Compute instances in the **Running** state are scaled in or out based on the configured auto scaling parameters. For compute instances in other states, only the configuration is saved, and the saved configuration takes effect when the compute instances are restarted.

  Table 11-56 Dynamic scaling parameters

Parameter	Description	Example Value
Load Collection Period	The interval for collecting instance load information, in seconds.	10
Scale-out Threshold	When the average value of the instance resource usage in the scale-in/out decision-making period exceeds the threshold, the instance starts to scale out.	0.9
Scale-out Size	The number of worker nodes to be added each time when the instance starts to scale out.	1
Scale-out Decision Period	The interval for determining whether to scale out an instance, in seconds.	200
Scale-out Timeout Period	The timeout period of the scale-out operation, in seconds.	400
Scale-in Threshold	When the average value of the instance resource usage in the scale-in/out decision-making period exceeds the threshold, the instance starts to scale in.	0.1
Scale-in Size	The number of worker nodes to be reduced each time when the instance starts to scale in.	1
Scale-in Decision Period	The interval for determining whether to scale in an instance, in seconds.	300
Scale-in Timeout Period	The timeout period of the scale-in operation, in seconds.	600

Step 4 After the configuration, click **OK**.

----End

11.10.3 Configuring a HetuEngine Maintenance Instance

Scenario

A maintenance instance is a special compute instance that performs automatic tasks. Maintenance instances are used to automatically refresh, create, and delete materialized views.

In a cluster, only one compute instance can be set as a maintenance instance, and the maintenance instance can also carry original computing services at the same time. If a tenant has multiple compute instances, only one compute instance can be used as the maintenance instance.

When you configure an existing compute instance as a maintenance instance, the status of the compute instance must be Stopped.

Procedure

Step 1 Log in to FusionInsight Manager as a user who can access the HetuEngine web UI.

Step 2 Choose **Cluster** > **Services** > **HetuEngine** to go its service page.

Step 3 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 4 In the **Compute Instance** page, locate the row that contains the tenant to which the target instance belongs and click **Configure** in the **Operation** column.

Step 5 Check whether **Maintenance Instance** in **Advanced Configuration** is set to **Yes**. If not, change the value to **Yes**.

Step 6 Set **Start Now** to **Yes** and click **OK**.

----End

11.10.4 Configuring the Nodes on Which HetuEngine Coordinator Is Running

By default, coordinator and worker nodes randomly start on Yarn NodeManager nodes, and you have to open all ports on all NodeManager nodes. Using resource labels of Yarn, HetuEngine allows you to specify NodeManager nodes to run coordinators.

Prerequisites

You have created a user for accessing the HetuEngine web UI. For details, see **Creating a HetuEngine Permission Role**.

Procedure

Step 1 Log in to FusionInsight Manager as a user who can access the HetuEngine web UI.

Step 2 Set Yarn parameters to specify the scheduler to handle PlacementConstraints.

1. Choose **Cluster** > **Services** > **Yarn**. Click the **Configurations** tab and then **All Configurations**. On the displayed page, search for

yarn.resourcemanager.placement-constraints.handler, set **Value** to **scheduler**, and click **Save**.

2. Click the **Instance** tab, select the active and standby ResourceManager instances, click **More**, and select **Restart Instance** to restart the ResourceManager instances of Yarn. Then wait until they are restarted successfully.

Step 3 Configure resource labels.

1. Choose **Tenant Resources** > **Resource Pool**. On the displayed page, click **Add Resource Pool**.
2. Select a cluster, and enter a resource pool name and a resource label name, for example, **pool1**. Select the desired hosts, click >> to add the selected hosts to the new resource pool, and click **OK**.

Step 4 Configure the queue capacity policy of a resource pool.

1. In the navigation pane on the left, click **Dynamic Resource Plan**. In the **Resource Distribution Policy** tab, select the resource pool created in the previous step for **Resource Pool**.
2. Locate the row that contains the target resource name in the **Resource Allocation** area, and click **Modify** in the **Operation** column.
3. In the **Modify Resource Allocation** dialog box, set the resource capacity policy for a queue in the selected resource pool. Ensure that **Maximum Resource** is greater than 0. For details, see **Configuring the Queue Capacity Policy of a Resource Pool**.

Step 5 Set HetuEngine parameters to enable the coordinator placement policy and enter the node resource label.

1. Choose **Cluster** > **Service** > HetuEngine. Click the **Configurations** tab and then **All Configurations**. On the displayed page, set parameters and click **Save**.

   Table 11-57 Setting HetuEngine parameters

Parameter	Setting
yarn.hetuserver.engine.coordinator.placement.enabled	true
yarn.hetuserver.engine.coordinator.placement.label	Node resource label created in **Step 3**, for example, **pool1**

2. Click **Dashboard**, click **More**, and select **Restart Service**. Wait until the HetuEngine service is restarted successfully.

Step 6 Restart the HetuEngine compute instance.

1. In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.
2. Stop the running compute instance and click **Start** in the **Operation** column to start the HetuEngine compute instance.

Step 7 Check the node on which the coordinator is running.

1. Return to FusionInsight Manager.
2. Choose **Cluster** > **Services** > **Yarn**. In the **Basic Information** area on the **Dashboard** page, click the link next to **ResourceManager WebUI**.
3. In the navigation pane on the left, choose **Cluster** > **Nodes**. You can view that the coordinator has been started on the node in the resource pool created in **Step 3**.

Figure 11-6 coordinator

----End

11.10.5 Importing and Exporting HetuEngine Compute Instance Configurations

On the HetuEngine web UI, you can import or export the instance configuration file and download the instance configuration template.

Procedure

Step 1 Log in to FusionInsight Manager as a user who can access the HetuEngine web UI and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 2 In the **Basic Information** area on the **Dashboard** tab page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 3 Click **Compute Instance**.

- Importing an instance configuration file: Click **Import**, select an instance configuration file in JSON format from the local PC, and click **Open**.

  NOTICE

  The import and export functions save only the configuration of compute instances. The instance ID, name, start time, end time, and status are not saved. After the import is complete, the information is generated again.

- Exporting an instance configuration file: Select the instances to be exported and click **Export** to export the current instance configuration file to the local PC.

----End

11.10.6 Viewing the HetuEngine Instance Monitoring Page

On the HetuEngine web UI, you can view the detailed information about a specified service, including the execution status of each SQL statement.

Procedure

Step 1 Log in to FusionInsight Manager as an administrator who can access the HetuEngine web UI and choose **Cluster** > *Name of the desired cluster* > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 2 In the **Basic Information** area on the **Dashboard** tab page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 3 Click **Compute Instance** then the tenant name corresponding to the instance to which operations are to be performed.

Step 4 Click **LINK** in the **WebUI** column to access the compute instance task monitoring page. If you access the CLUSTER OVERVIEW page for the first time, you can view information on the compute instance task monitoring page.

Table 11-58 Metric description

Metric	Description
Running Queries	Indicates the number of tasks concurrently executed on the current instance.
Active Workers	Indicates the number of valid worker nodes on the current instance.
ROWS/SEC	Indicates the number of data rows processed by the current instance per second.
Queued Queries	Indicates the number of tasks to be executed in the waiting queue on the current instance.
RUNNABLE DRIVERS	Indicates the number of running drivers on the current instance.
BYTES/SEC	Indicates the amount of data read from the current instance per second.
Blocked Queries	Indicates the number of blocked tasks on the current instance.
RESERVED MEMORY (B)	Indicates the memory occupied by running tasks on the current instance.
WORKER PARALLEISM	Indicates the average CPU time slice used by each worker on the current instance per second.
Avg CPU cycles per worker	Indicates the average CPU cycles of each worker node of the current instance.

Step 5 Filter query tasks by **State** on the **QUERY DETAILS** page.

Table 11-59 State description

State	Description
Running	Views running tasks.
Queued	Views the tasks to be executed in the waiting queue.
Finished	Views the finished tasks.
Failed	Views failed tasks, which can be filtered by failure cause.

Step 6 Click a task ID to view the basic information, resource usage, stages, and tasks. For a failed task, you can view related logs on the detail query page.

Figure 11-7 Viewing task details

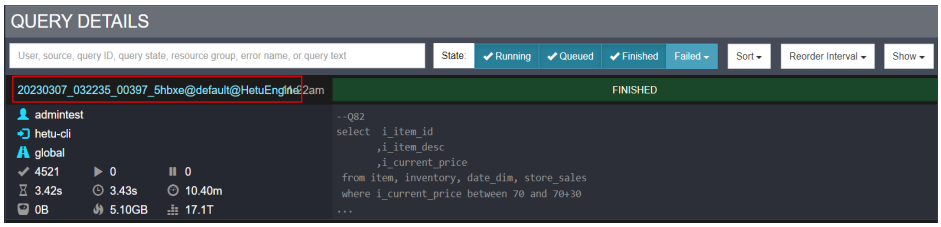


Figure 11-8 Task resource utilization summary

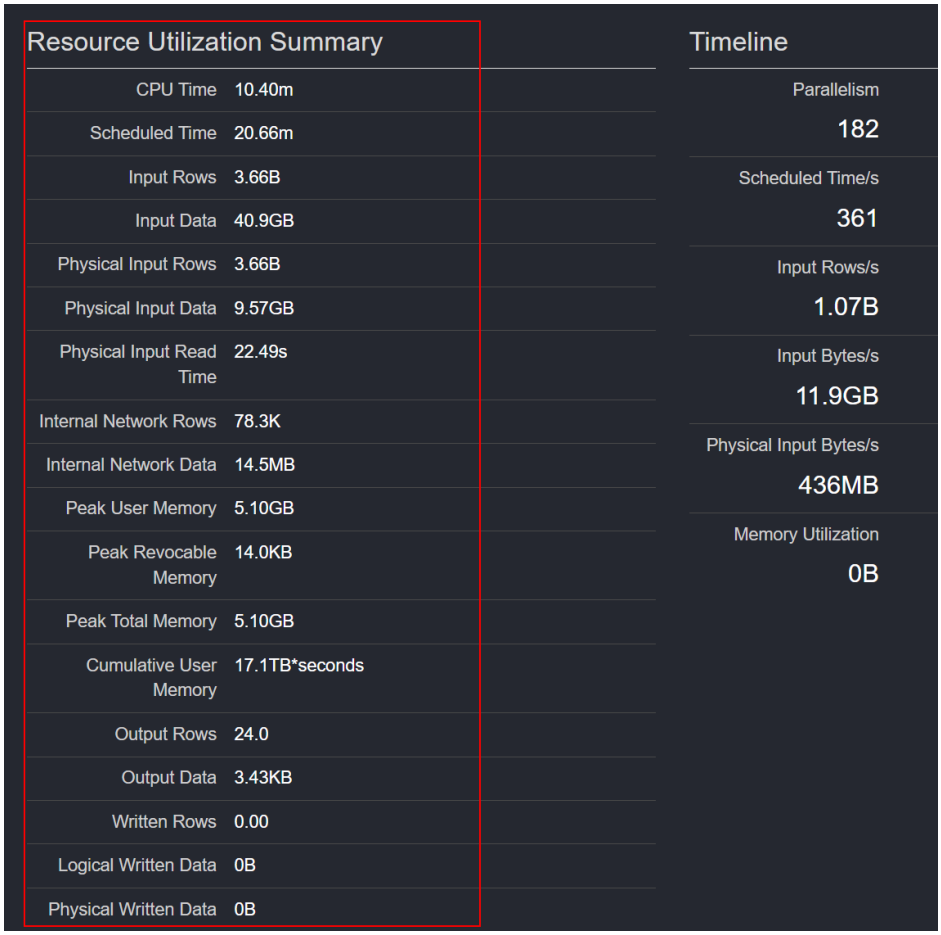


Figure 11-9 Stages

Table 11-60 Stages monitoring information

Monitoring Item	Description
SCHEDULED TIME SKEW	Indicates the scheduled time of the concurrent tasks on a node in the current stage.
CPU TIME SKEW	Indicates whether concurrent tasks have computing skew in any stage phase.

Figure 11-10 Tasks (Clicking the triangle on the right of each stage to view the tasks)

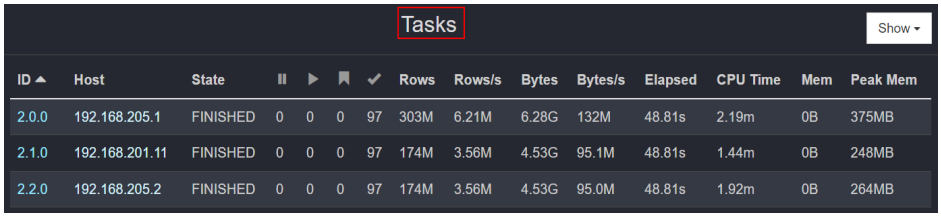


Table 11-61 Tasks monitoring items

Monitoring Item	Description
ID	Indicates the ID of the task that is concurrently executed in multiple phases. The format is *Stage ID*:*Task ID*.
Host	Indicates the Worker node where the current task is being executed.
State	Indicates the task execution status, including **PLANNED**, **RUNNING**, **FINISHED**, **CANCELED**, **ABORTED**, and **FAILED**.
Rows	Indicates the total number of data records read by a task. The unit is thousand (k) or million (M). By analyzing the number of data records read by different tasks in the same stage, you can quickly determine whether data skew occurs in the current task.
Rows/s	Indicates the number of data records read by a task per second. By analyzing the number of data records read by different tasks in the same stage, you can quickly determine whether the network bandwidth of the node is different and whether the node NIC is faulty.
Bytes	Indicates the data volume read by a task.
Bytes/s	Indicates the data volume read by a task per second.
Elapsed	Indicates the task execution duration.

Monitoring Item	Description
CPU Time	Indicates the CPU time used by a task.
Mem	Task memory
Peak Mem	Peak memory usage of tasks

Step 7 Click the "Host" link to view the task resource usage of each node.

Figure 11-11 Resource usage of the Task node

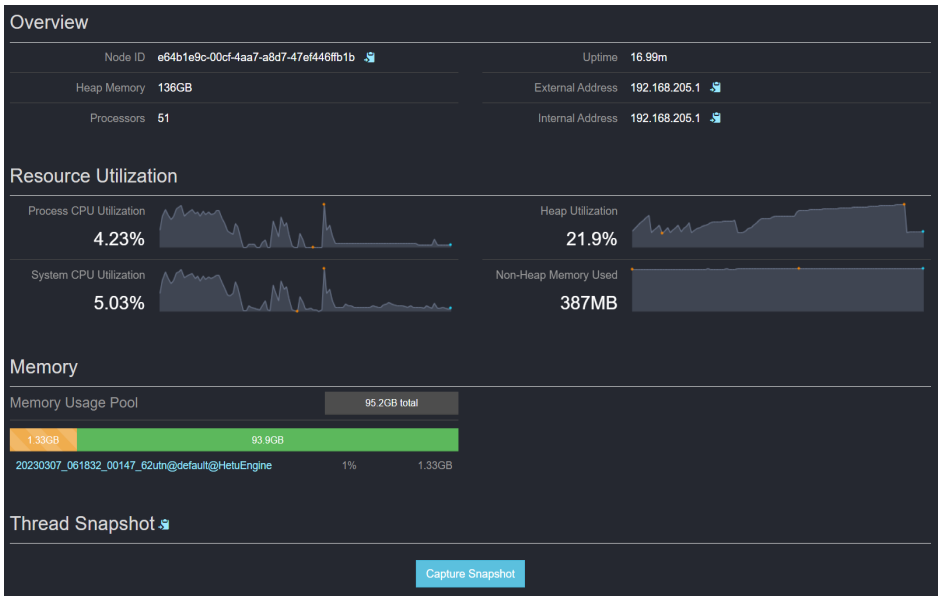


Table 11-62 Monitoring metrics of node resources

Name	Description
Node ID	Indicates the host ID.
Heap Memory	Indicates the maximum heap memory size.
Processors	Indicates the number of processors.
Uptime	Indicates the running duration.
External Address	Indicates the external IP address.
Internal Address	Indicates the internal IP address.
Process CPU Utilization	Indicates the physical CPU utilization.
System CPU Utilization	Indicates the system CPU utilization.
Heap Utilization	Indicates the heap memory utilization.

Name	Description
Non-Heap Memory Used	Indicates the non-Heap memory size.
Memory Usage Pool	Indicates the memory pool size of the current Worker node.

----End

11.10.7 Viewing HetuEngine Coordinator and Worker Logs

On the HetuEngine web UI, you can click the LogUI link to go to the YARN web UI and view Coordinator and Worker logs.

Step 1 Log in to FusionInsight Manager as a user who can access the HetuEngine web UI and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 2 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 3 Click **Compute Instance** and select the compute instance on which operations are to be performed under the corresponding tenant. Click **Coordinator** or **Worker** in the **LogUI** column to view coordinator or worker logs on the YARN web UI.

----End

11.10.8 Configuring HetuEngine Query Fault Tolerance

This section applies to MRS 3.3.0 or later.

Scenario

When a node in the cluster is faulty due to network, hardware, or software problems, all query tasks running on the node are lost. This seriously affects cluster productivity and wastes resources, especially for queries running for a long time. HetuEngine provides a fault recovery mechanism, that is, the fault tolerance execution capability. The cluster can reduce the probability of query failure by automatically re-running affected queries or their component tasks. This reduces manual intervention and improves fault tolerance, but prolongs the total execution time.

Currently, the following fault tolerance execution mechanisms are supported:

- Query-level retry policy: If query-level fault tolerance is enabled, intermediate data will not be flushed to disks. If a query job fails, all tasks of the query job will be automatically retried. This policy is recommended when most of the cluster's workloads are small queries.
- Task-level retry policy: If task-level fault tolerance is enabled, HDFS is configured as the swap area by default to flush exchange intermediate data to disks. If a query job fails, the failed tasks are retried. You are advised to use this policy when performing a large number of queries. In this way, the cluster can efficiently retry small-granularity tasks in the query instead of the entire query.

This example describes how to set the fault tolerance execution mechanism of the task-level retry policy.

Notes

- Fault tolerance does not apply to corrupted queries or other user error scenarios. For example, resources are not spent retrying query tasks that fail because SQL statements cannot be parsed.
- Different data sources have different fault tolerance capabilities for SQL statements.
  - All data sources support fault-tolerant execution of **read operations**.
  - Hive data sources are fault-tolerant of write operations.
- This tolerance function is good for large-scale queries. If you run a large number of short small queries on a fault-tolerant cluster at the same time, a latency may occur. Therefore, it is recommended that you use dedicated fault-tolerant compute instances when processing batch operations, which are isolated from compute instances with higher query volume for interactive queries.

Procedure

Step 1 Log in to FusionInsight Manager as a user who can access the HetuEngine web UI and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 2 In the **Basic Information** area on the **Dashboard** tab page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 3 On the **Compute Instance** tab, locate the row containing the tenant to which the desired instance belongs and click **Configure** in the **Operation** column.

Step 4 In the **Custom Configuration** area, click **Add** to add the following parameters:

Table 11-63 Fault tolerance execution parameters

Parameter	Example Value	Configuration File	Description
retry-policy	TASK	• coordinator.config.properties • worker.config.properties	• Retry policy for fault tolerance execution. • Value range: **QUERY** and **TASK**
task-retry-attempts-per-task	4	• coordinator.config.properties • worker.config.properties	• Maximum number of attempts to retry a single task before a query failure is declared when task fault tolerance is enabled. • Default value: **4**

Parameter	Example Value	Configuration File	Description
query-retry-attempts	4	• coordinator.config.properties • worker.config.properties	• Maximum number of attempts to retry a single query before a query failure is declared when query fault tolerance is enabled. • Default value: **4**
fault-tolerant-execution-task-memory	5GB	• coordinator.config.properties • worker.config.properties	• This parameter is available when **retry-policy** is set to **TASK**. If this parameter is not set, the default value **5** GB is used. The node allocates tasks based on the available memory and estimated memory usage. • This parameter is used to estimate the memory required for initial task allocation. A larger value indicates that each task uses more memory but the cluster concurrency capability decreases. You can dynamically adjust the value based on service requirements.

Step 5 Set **Start Now** to **Yes** and click **OK**.

NOTICE

- After task-level fault tolerance is enabled, intermediate data is generated and cached in the file system. A large number of concurrent queries cause great disk pressure on the file system. By default, HetuEngine can buffer intermediate data to the temporary directory in HDFS. When OBS is connected in the scenario where storage and compute are decoupled, task-level fault tolerance is supported, but intermediate data is still flushed to the disk of the HDFS temporary directory.
- By default, the cluster clears buffer files when the query is complete, and checks and clears residual buffer files that have expired for one day every hour. You can perform the following operations to disable the periodic clearing function:

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HetuEngine**, click **Configurations** then **All Configurations**, click **HSBroker(Role)**, select **Fault-tolerance execution**, set **fte.exchange.clean.task.enabled** to **false**, and save the configuration. Click **Instance**, select all HSBroker instances, click **More**, select **Restart Instance**, and restart the instances as prompted for the configuration to take effect.

----End

11.11 HetuEngine Performance Tuning

11.11.1 Adjusting YARN Resource Allocation

HetuEngine depends on the resource allocation and control capabilities provided by Yarn. You need to adjust the Yarn service configuration based on the actual service and cluster server configuration to achieve the optimal performance.

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **Yarn** > **Configurations** > **All Configurations** and set Yarn service parameters by referring to **Table 11-64**.

Table 11-64 Yarn configuration parameters

Parameter	Description	Default Value	Recommended Value
yarn.nodemanager.resource.memory-mb	Total physical memory on the node that can be used by Yarn. The default value is 16,384 MB. If the node has permanent processes of other services, reduce this value to reserve sufficient resources for the processes.	16384	To achieve the optimal performance, set this parameter to 90% of the minimum physical memory of the node in the cluster.

Parameter	Description	Default Value	Recommended Value
yarn.nodemanager.resource.cpu-vcores	Number of CPU cores that can be allocated to a container.	8	To achieve the optimal performance, set this parameter to the minimum number of vCores of the node in the cluster.
yarn.scheduler.maximum-allocation-mb	Maximum memory requested by each container of ResourceManager. The unit is MB. If much memory is requested, the memory that is set by the parameter is allocated.	65536	To achieve the optimal performance, set this parameter to 90% of the minimum physical memory of the node in the cluster.
yarn.scheduler.maximum-allocation-vcores	Maximum value requested by each container of ResourceManager, represented by the number of virtual CPU cores. Requests where the requested values are greater than this value are invalid and the values will be overwritten by this parameter.	32	To achieve the optimal performance, set this parameter to the minimum number of vCores of the node in the cluster.

Step 3 Click **Save**.

Step 4 Choose **Cluster** > **Services** > **Yarn** > **More** > **Restart Service** to restart the Yarn service for the parameters to take effect.

----End

11.11.2 Adjusting HetuEngine Cluster Node Resource Configurations

The default memory size and disk overflow path of HetuEngine are not the best. You need to adjust node resources in the cluster based on the actual service and server configuration of the cluster to achieve the optimal performance.

Adjusting HetuEngine Cluster Node Resource

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **HetuEngine** > **Configurations** > **All Configurations** and adjust the cluster node resource parameters by referring to **Table 11-65**.

Table 11-65 Parameters for configuring cluster node resources

Parameter	Default Value	Recommended Value	Description	Parameter File
yarn.hetuserver.engine.coordinator.memory	5120	At least 2 GB less than that of **yarn.scheduler.maximum-allocation-mb**	Memory size used by a Coordinator node	application.properties
yarn.hetuserver.engine.coordinator.number-of-containers	2	2	Number of Coordinator nodes	application.properties
yarn.hetuserver.engine.coordinator.number-of-cpus	1	At least two vCores less than **yarn.scheduler.maximum-allocation-vcores**	CPU vCores used by a Coordinator node	application.properties
yarn.hetuserver.engine.worker.memory	10240	At least 2 GB less than that of **yarn.scheduler.maximum-allocation-mb**	Memory size used by a worker node	application.properties
yarn.hetuserver.engine.worker.number-of-containers	2	Adjusted based on application requirements	Number of worker nodes	application.properties
yarn.hetuserver.engine.worker.number-of-cpus	1	At least two vCores less than **yarn.scheduler.maximum-allocation-vcores**	CPU vCores used by a Worker node	application.properties
Xmx size in the **extraJavaOptions** parameter	8 GB	*Memory size used by a worker node x 0.8*	Maximum available memory of the worker JVM process	worker.jvm.config
query.max-memory-per-node	5 GB	Worker JVM x 0.7	Maximum available memory of a Query node	worker.config.properties
query.max-total-memory-per-node	5 GB	Worker JVM x 0.7	Maximum available memory of a Query + System node	worker.config.properties

Parameter	Default Value	Recommended Value	Description	Parameter File
memory.heap-headroom-per-node	3 GB	Worker JVM x 0.3	Maximum available memory of a system heap node	worker.config.properties
Xmx size in the **extraJavaOptions** parameter	4 GB	*Memory size used by a Coordinator node x 0.8*	Maximum available memory of the Coordinator JVM process	coordinator.jvm.config
query.max-memory-per-node	3 GB	Coordinator JVM x 0.7	Maximum memory that can be used for node query	coordinator.config.properties
query.max-total-memory-per-node	3 GB	Coordinator JVM x 0.7	Maximum available memory of a Query + System node	coordinator.config.properties
memory.heap-headroom-per-node	1 GB	Coordinator JVM x 0.3	Maximum available memory of a system heap node	coordinator.config.properties
query.max-memory	7 GB	Sum(query.max-memory-per-node) x 0.7	Maximum available memory of a Query cluster	worker.config.properties/ coordinator.config.properties
spiller-spill-path	CONTAINER_ROOT_PATH/tmp/ hetuserver/ hetuserver-sqlengine/	One or more independent SSDs	Disk output file path	worker.config.properties/ coordinator.config.properties

Parameter	Default Value	Recommended Value	Description	Parameter File
max-spill-per-node	10 GB	Sum(Available space of each node) x 50%	Available disk space for storing output files of all queries on a node	worker.config.properties/ coordinator.config.properties
query-max-spill-per-node	10 GB	80% of the available disk space on a node	Available disk space for storing output files of a query on a node	worker.config.properties/ coordinator.config.properties

Step 3 Click **Save**.

Step 4 Choose **Cluster** > **Services** > **HetuEngine** > **More** > **Restart Service** to restart the HetuEngine service for the parameters to take effect.

Step 5 Restart the HetuEngine compute instance that is running.

1. Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**.
2. In the displayed**Dashboard** tab, find the **Basic Information** area, and click the link next to **HSConsole WebUI**.
3. Click **Compute Instances**, select the running HetuEngine instance, click **Restart**, and operate as prompted.

----End

11.11.3 Optimizing HetuEngine INSERT Statements

You can add custom configurations based on the number of partition columns in the query result to achieve optimal writing performance when using HetuEngine to write data to a Hive data source partition table.

Procedure

Step 1 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 2 Click **Configurations** and then **All Configurations**. Search for **task.writer-count** (number of concurrent writer threads for a single query on each worker), and ensure the value is **1**.

Step 3 In the **Basic Information** area on the **Dashboard** page, click the link next to **HSConsole WebUI**.

Step 4 On HSConsole, click **Data Source**. Locate the row that contains the target Hive data source, click **Edit** in the **Operation** column, and add custom configurations. You can adjust custom parameters by referring to **Table 11-66**.

Table 11-66 Performance optimization parameters of the INSERT statement

Parameter	Description
hive.max-partitions-per-writers	The value must be greater than or equal to the product of the values generated by the **Count(distinct)** method for all partition columns of the Hive data source partition table to which data is to be written.

NOTE

The following is an example of the **Count(distinct)** method:

The **t2** table contains the **col1**, **col2**, and **col3** columns. The query result is as follows:

col1 col2 col3

A 100 5

C 103 4

B 101 3

E 110 4

D 100 5

- If **col3** is a partition column and its **Count(distinct)** value is **3**, you are advised to set **hive.max-partitions-per-writers** to a value no less than **3**.
- If the result table has multiple partition columns, for example, **col2** and **col3**, and the **Count(distinct)** values of **col2** and **col3** are **4** and **3**, respectively, you are advised to set **hive.max-partitions-per-writers** to a value no less than **12**.

Step 5 Click **OK**.

----End

11.11.4 Adjusting HetuEngine Metadata Caching

Scenario

When HetuEngine accesses the Hive data source, it needs to access the Hive metastore to obtain the metadata information. HetuEngine provides the metadata cache function. When the database or table of the Hive data source is accessed for the first time, the metadata information (database name, table name, table field, partition information, and permission information) of the database or table is cached, the Hive metastore does not need to be accessed again during subsequent access. If the table data of the Hive data source does not change frequently, the query performance can be improved to some extent.

Procedure

Step 1 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**.

Step 2 On the **Dashboard** tab page that is displayed, find the **Basic Information** area, and click the link next to **HSConsole WebUI**.

Step 3 On HSConsole, click **Data Source**. Locate the row that contains the target Hive data source, click **Edit** in the **Operation** column, and add custom configurations according to **Table 11-67**.

Table 11-67 Metadata cache parameters

Parameter	Description	Default Value
hive.metastore-cache-ttl	Cache duration of the metadata of the co-deployed Hive data source.	0s
hive.metastore-cache-maximum-size	Maximum cache size of the metadata of the co-deployed Hive data source.	10000
hive.metastore-refresh-interval	Interval for refreshing the metadata of the co-deployed Hive data source.	1s
hive.per-transaction-metastore-cache-maximum-size	Maximum cache size of the metadata for each transaction of the co-deployed Hive data source.	1000

Step 4 Click **OK**.

Step 5 Restart the HetuEngine service.

Log in to the Manager. On the **Dashboard** page, click **More** and select **Restart Service** to restart the HetuEngine service as prompted.

Step 6 Restart the HetuEngine compute instance that is running.

On HSConsole, click **Compute Instances**, select the running HetuEngine instance, click **Restart**, and operate as prompted.

----End

11.11.5 Enabling Dynamic Filtering in HetuEngine

HetuEngine provides the dynamic filtering function, which significantly improves the performance in join scenarios.

This section describes how to enable dynamic filtering.

Step 1 Log in to FusionInsight Manager as a user who can access the HetuEngine web UI and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

Step 2 In the **Basic Information** area on the **Dashboard** tab page, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

Step 3 In the **Compute Instance** page, locate the row that contains the tenant to which the target instance belongs and click **Configure** in the **Operation** column.

Step 4 In the **Custom Configuration** area, click **Add** to add the following parameters:

Table 11-68 Dynamic filtering parameters

Parameter	Value	ConFile	Parameter Description
enable-dynamic-filtering	true	**coordinator.config.properties** and **worker.config.properties**	Whether to enable the dynamic filtering function. The default value is **false**.

Step 5 Set **Start Now** to **Yes** and click **OK**.

----End

11.11.6 Adjusting the Execution of Adaptive Queries in HetuEngine

This section applies to MRS 3.2.0 or later.

Scenario

Typically, SQL statements of large tasks (for example, scanning large amounts of data from an entire table) occupy a large number of resources, which affects the load of other tasks in case that resources are insufficient. This deteriorates user experience and increases O&M costs. To resolve this issue, HetuEngine provides the adaptive query execution function to allow adaptive scheduling and execution of queries.

This section describes how to enable adaptive query execution.

Procedure

Step 1 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**.

Step 2 On the **Dashboard** tab page that is displayed, find the **Basic Information** area, and click the link next to **HSConsole WebUI**.

Step 3 On the page that is displayed, click **Data Source**, locate the row containing the Hive data source to be modified, and click **Edit** in the **Operation** column.

Step 4 Add **hive.strict-mode-restrictions** to the custom parameters and set it to **NONE** to enable the adaptive query execution function. For details, see **Step 6.7**.

Step 5 Click **OK**.

Step 6 Restart the HetuEngine service.

Log in to the Manager. On the **Dashboard** page, click **More** and select **Restart Service** to restart the HetuEngine service as prompted.

Step 7 Restart the HetuEngine compute instance that is running.

On HSConsole, click **Compute Instances**, select the running HetuEngine instance, click **Restart**, and operate as prompted.

----End

11.11.7 Adjusting Timeout for Hive Metadata Loading

A large partitioned table contains too many partitions. As a result, the task times out. In addition, a large number of partitions may take more time to load and synchronize with the metadata storage cache. To achieve better performance in larger-scale storage, you are advised to adjust the maximum timeout interval for loading the metadata cache and the maximum waiting time for loading the metadata connection pool accordingly.

Step 1 Log in to FusionInsight Manager as a HetuEngine administrator and choose **Cluster** > **Services** > **HetuEngine**.

Step 2 On the **Dashboard** tab page that is displayed, find the **Basic Information** area, and click the link next to **HSConsole WebUI**.

Step 3 Click **Data Source**, locate the row that contains the Hive data source, click **Edit** in the **Operation** column, and add the following custom parameters:

Table 11-69 Metadata timeout parameters

Parameter	Default Value	Description
hive.metastore-timeout	10s	• Specifies the maximum timeout interval (in seconds or minutes) for caching metadata loaded by the Hive data source in co-deployment scenarios. • For operations in a large partition table, the value can be 60s or greater. Set this parameter based on the data volume.
hive.metastore.connection.pool.maxWaitMillis	1000	• Specifies the maximum waiting time of the connection pool (in milliseconds) for loading metadata to the Hive data source in co-deployment scenarios. • If the connection pool is frequently accessed and the number of connections in the connection pool is small, the value can be 100000 or larger. Set this parameter based on the service volume.

Step 4 Click **OK**.

----End

11.11.8 Tuning Hudi Data Source Performance

This section applies to MRS 3.3.1 or later.

HetuEngine can access data sources such as Hive and Hudi at a high speed. Hudi data source optimization includes Hudi table design optimization and cluster environment optimization.

Hudi Table Tuning

You can optimize table and data structures by referring to the following suggestions:

- Partition tables by the fields frequently used as filter conditions.
- If there mostly are equivalent queries with primary keys or primary key subsets, use bucket indexes to create tables and use query fields as bucketing keys.
- When querying a MOR table, periodically compact data to improve query performance. For details, see **COMPACTION**.

Cluster Environment Optimization

You can adjust the YARN configuration, cluster node resource configurations, metadata cache, and dynamic filter policies to optimize the system.

- For details about how to adjust the YARN configuration, see **Adjusting YARN Resource Allocation**.
- To adjust cluster node resource configurations, see **Adjusting HetuEngine Cluster Node Resource Configurations**.
- To adjust the metadata cache configuration, see **Adjusting HetuEngine Metadata Caching**.
- To adjust the dynamic filter policy, see **Enabling Dynamic Filtering in HetuEngine**.

Example

A user stores device order information in a Hudi MOR table. The user can query the order details by order number. The number of orders per day is stable, and there are small peak hours during holidays.

- The order number is unique. In more than 80% queries, the order number is used for equivalent queries. The SQL statement is similar to **select * from table where order_id = 'id1';**.
- You can use day as the partition key since the number of orders per day is stable.
- Historical partition updates are not frequent, and main data is updated in new partitions.

Optimization suggestions

1. Use bucket indexes to create tables with Spark-SQL. The index key is the order ID, and the partition key is the date.
2. Compact logs periodically to improve query performance.

The following are example SQL statements:

```
set hoodie.compact.inline=true;
set hoodie.schedule.compact.only.inline=true;
```



```
set hoodie.run.compact.only.inline=false;
create table hudi_mor (order_id int, comb int, col1 string, col2 string, dt int)
using hudi
partitioned by(dt)
options(type='mor', primaryKey='order_id', preCombineField='comb',
hoodie.index.type = 'BUCKET',
hoodie.bucket.index.num.buckets=100,
hoodie.bucket.index.hash.field = 'order_id')
```

11.12 HetuEngine Log Overview

Log Description

Log paths

The HetuEngine logs are stored in **/var/log/Bigdata/hetuengine/** and **/var/log/Bigdata/audit/hetuengine/**.

Log archiving rules

Log archiving rules use the FixedWindowRollingPolicy policy. The maximum size of a single file and the maximum number of log archive files can be configured. The rules are as follows:

- When the size of a single file exceeds the default maximum value, a new compressed archive file is generated. The naming rule of the compressed archive log file is as follows: *<Original log name>.[ID]*.log.gz.
- Log deletion rules
  - When the total size of compressed run logs of the HetuEngine compute instance reaches the maximum value, the earliest log file is deleted.

    Run logs of HetuEngine compute instances are synchronized to HDFS and retained for 30 days (**log.clean.task.expire-time.day**) by default. The archive path is **hdfs://hacluster/hetuserverhistory/tenant/coordinator**.
  - When the number of other archived log files reaches the maximum value or the total size of compressed log files reaches the maximum value, the earliest log files are deleted.

By default, the maximum size of an audit log file is 30 MB, and the maximum number of log archive files is 20.

By default, the maximum size of a single run log file is 100 MB, and the maximum number of archived log files is 20. The maximum size of a single HetuEngine compute instance run log file is 100 MB, and logs archived in HDFS are retained for 30 days by default.

To change the maximum size of a single run log file or audit log file or change the maximum number of log archive files of an instance, perform the following operations:

Step 1 Log in to Manager.

Step 2 Choose **Cluster** > **Services** > **HetuEngine** > **Configurations** > **All Configurations**.

Step 3 In the parameter list of log levels, search for **logback.xml** to view the current run log and audit log configurations of HSBroker, HSConsole, HSFabric, and QAS.

NOTE

Parameters related to HetuEngine compute instance run logs

- **log.clean.task.enabled**: indicates whether to enable scheduled compute instance log clearing.
- **log.clean.task.expire-time.day**: indicates the expiration time of compute instance logs archived in HDFS. The default value is 30 days.
- **log.max-history**: indicates the maximum retention period of compute instance logs locally. The default value is 7 days.
- **log.clean.task.schedule.plan**: indicates the schedule for automatically clearing compute instance logs. The value is a cron expression. Only a fixed triggering time in a day can be specified.
- **log.max-size**: indicates the maximum size of a single log file of a HetuEngine compute instance. The default value is 100 MB.
- **log.max-total-size**: indicates the maximum size of compressed HetuEngine compute instance log files. The default value is 5 GB.

Step 4 Select the configuration item to be modified and modify it.

Step 5 Click **Save**, and then click **OK**. The configuration automatically takes effect after about 30 seconds.

----End

Table 11-70 HetuEngine log list

Log Category	Log File	Description
Installation, startup, and stop log	prestart.log	Preprocessing script log before startup
	start.log	Startup log
	stop.log	Stop log
	postinstall.log	Installation log
Run log	*Instance name*.log	Run log
	*Instance name*_wsf.log	Interface parameter verification log
	hdfs://hacluster/hetuserverhistory/*Tenant*/*Coordinator or worker*/*application_ID*/*container_ID*/*yyyyMMdd*/server.log	Run log of the HetuEngine compute instance
Status check log	service_check.log	Health check log
	service_getstate.log	Status check log
	availability-check.log	HetuEngine status check log

Log Category	Log File	Description
	haCheck.log	Log generated when the QAS checks the HA status
Audit log	*Instance name*-audit.log	Audit log
	hsbroker-audit.log	Audit log of HSBroker operations
	hsconsole-audit.log	Audit log of HSConsole operations
	hsfabric-audit.log	Audit log of HetuEngine operations performed across domains
	hdfs://hacluster/hetuserverhistory/*Tenant*/coordinator/*application_ID*/*container_ID*/*yyyyMMdd*/hetuserver-engine-audit.log	Audit log of the HetuEngine compute instance
queryInfo log	hdfs://hacluster/hetuserverhistory/*Tenant*/*Coordinator*/*application_ID*/*container_ID*/*yyyyMMdd*/queryinfo.log	queryInfo log of HetuEngine compute instances, which records SQL running statistics.
Clean log	cleanup.log	Script cleanup log
Initialization log	hetupg.log	Metadata initialization log
	ranger-trino-plugin-enable.log	Operation log of integrating Ranger plug-ins to the HetuEngine kernel
Client log	qas_client.log	ZooKeeper client log of the QAS instance
Stack information log	threadDump-<*DATE*>.log	Log printed when instances are restarted or stopped
Other	hetu-updateKrb5.log	Log generated when the Hive data source configuration is automatically updated after the Hive cluster domain is changed.

Log Category	Log File	Description
	hetu_utils.log	Log generated when the preprocessing script calls the tool class to upload files to the HDFS during startup.

Log Level

Table 11-71 describes the log levels provided by HetuEngine. The priorities of log levels are OFF, ERROR, WARN, INFO, and DEBUG in descending order. Logs whose levels are higher than or equal to the specified level are printed. The number of printed logs decreases as the specified log level increases.

Table 11-71 Log levels

Level	Description
OFF	Logs of this level record no logs.
ERROR	Logs of this level record error information about the current event processing.
WARN	Logs of this level record exception information about the current event processing.
INFO	Logs of this level record normal running status information about the system and events.
DEBUG	Logs of this level record the system information and system debugging information.

To change the run log or audit log level of an instance, perform the following steps:

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **HetuEngine** > **Configurations** > **All Configurations**.

Step 3 In the parameter list of log levels, search for **logback.xml** to view the current run log and audit log levels of HSBroker, HSConsole, and HSFabric.

Step 4 Select a desired log level.

Step 5 Click **Save**, and then click **OK**. The configuration automatically takes effect after about 30 seconds.

----End

To change the HetuEngine Coordinator/Worker log level, perform the following steps:

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **HetuEngine** > **Configurations** > **All Configurations**.

Step 3 In the parameter list of log levels, search for **log.properties** to view the current log levels.

Step 4 Select a desired log level.

Step 5 Click **Save**, and then click **OK**. Wait until the operation is successful.

Step 6 Choose **Cluster** > **Services** > **HetuEngine** > **Instance**, click the HSBroker instance in the role list, and choose **More** > **Restart Instance**.

Step 7 After the HSBroker instance is restarted, choose **Cluster** > **Services** > **HetuEngine**. On the overview page, click the link next to **HSConsole WebUI** to go to the compute instance page.

Step 8 Select the compute instances to be restarted and click **Stop**. After all instances are stopped, click **Start** to restart the compute instances.

----End

Log Format

The following table lists the HetuEngine log formats.

Table 11-72 Log format

Log Type	Format	Example
Run log	<*yyyy-MM-dd HH:mm:ss,SSS*>|<*Log level*>|<*Name of the thread that generates the log*>|<*Message in the log*>|<*Location where the log event occurs*>	2024-05-22 06:03:35,696 | INFO | main | Construct zooKeeper helper finished. | com.xxx.hetuserver.hsbroker.core.zookeeper.ZooKeeperClient (ZooKeeperClient.java:312)
Audit log	<yyyy-MM-dd HH:mm:ss,SSS>|<Log level>|<Name of the thread that generates the log>|UserName=<*User name*>UserIP=<*User IP*>Time=<*Event time*>Operation=<*Operation*>Result=<*Operation result*>Detail=<*Details*> | xxx	2024-05-22 14:12:24,967 | INFO | https-jsse-nio-192.168.43.244-29860-exec-10 | UserName=hetuserver/hadoop.eef78bf6_bce3_47ff_b808_ec90ae6f6a2a.com@EEF78BF6_BCE3_47FF_B808_EC90AE6F6A2A.COM UserIP=192.168.43.244 Time=2024-05-22 14:12:24 Operation=Login stmt={kerberos login} Result=SUCCESS Detail=SUCCESS | audit xxx

11.13 Common HetuEngine SQL Syntax

11.13.1 HetuEngine Data Type

Currently, the following data types are supported during table creation: TINYINT, SMALLINT, BIGINT, INT, BOOLEAN, REAL, DECIMAL, DOUBLE, VARCHAR, STRING, BINARY, VARBINARY, TIMESTAMP, DATE, CHAR, ARRAY, ROW, MAP, and STRUCT. Other types are supported during data query and calculation.

Generally, most non-composite data types can be entered by literals and character strings. In the example, a string in the JSON format is added.

```
select json '{"name": "aa", "gender": "man"}';
              _col0
---------------------------------
 {"name":"aa","gender":"man"}
(1 row)
```

Boolean

The valid text values for "true" are **TRUE**, **t**, **true**, and **1**.

The valid text values for the "false" value are **FALSE**, **f**, **false**, and **0**.

TRUE and **FALSE** are standard usages (SQL-compatible).

The following is an example:

```
select BOOLEAN '0';
 _col0
-------
 false
(1 row)

select BOOLEAN 'TRUE';
 _col0
-------
 true
(1 row)

select BOOLEAN 't';
 _col0
-------
 true
(1 row)
```

Integer

Table 11-73 Integer

Parameter	Description	Storage Space	Value Range	Literal
TINYINT	Tiny integer	8 bits	-128~127	TINYINT
SMALLINT	Small integer	16 bits	-32,768 ~ +32,767	SMALLINT
INTEGER	Integer	32 bits	-2,147,483,648 ~ +2,147,483,647	INT

Parameter	Description	Storage Space	Value Range	Literal
BIGINT	Big integer	64 bits	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807	BIGINT

The following is an example:

```
--Create a table containing TINYINT data:
CREATE TABLE int_type_t1   (IT_COL1 TINYINT) ;
--Insert data of the TINYINT type:
insert into int_type_t1   values (TINYINT'10');
--View data:
SELECT * FROM int_type_t1;
 it_col1
---------
 10
(1 row)
--Drop a table:
DROP TABLE int_type_t1;
```

Fixed Precision

Parameter	Description	Storage Space	Value Range	Literal
DECIMAL	Decimal number with fixed precision. The maximum precision is 38 bits, but the precision with less than 18 bits can ensure the best performance. Decimal has two input parameters: • **precision**: total number of digits. The default value is **38**. • **scale**: number of decimal places. The default value is **0**. **NOTE** If **scale** is **0** and **precision** is **38**, the maximum precision is 19 bits.	64 bits	$-10^{38}+1 \sim 10^{38}-1$	DECIMAL
NUMERIC	Same as DECIMAL	128 characters	$-10^{38}+1 \sim 10^{38}-1$	NUMERIC

Table 11-74 Examples of literals

Example	Data Type
DECIMAL '0'	DECIMAL(1)
DECIMAL '12345'	DECIMAL(5)
DECIMAL '0000012345.1234500000'	DECIMAL(20, 10)

```
--Create a table containing DECIMAL data.
CREATE TABLE decimal_t1 (dec_col1 DECIMAL(10,3)) ;

– Insert data of the DECIMAL type.
insert into decimal_t1 values (DECIMAL '5.325');

--View data.
SELECT * FROM decimal_t1;
 dec_col1
---------
 5.325
(1 row)

-- Negative example: The number of decimal places exceeds the defined length. As a result, the SQL statement fails to be executed.
insert into decimal_t1 values (DECIMAL '5.3253');
Query 20201126_034601_00053_tq98i@default@HetuEngine failed: Insert query has mismatched column types: Table: [decimal(10,3)], Query: [decimal(5,4)]

--Drop the table.
DROP TABLE decimal_t1;

--Create a NUMERIC type table.
CREATE TABLE tb_numberic_hetu(col1 NUMERIC(9,7));
CREATE TABLE

--Insert data.
INSERT INTO tb_numberic_hetu values(9.12);
INSERT: 1 row

--View data.
SELECT * FROM tb_numberic_hetu;
col1
------------
 9.1200000
(1 row)
```

Float

Parameter	Description	Storage Space	Value Range	Literal
REAL	Real number	32 bits	1.40129846432481707e-45 to 3.40282346638528860e+38, positive or negative	REAL

Parameter	Description	Storage Space	Value Range	Literal
DOUBLE	Double-precision floating point number with 15 to 17 valid digits, depending on the application scenario. The number of valid digits does not depend on the decimal point.	64 bits	4.94065645841246544e-324 to 1.79769313486231570e+308, positive or negative	DOUBLE
FLOAT	Single-precision floating point number with 6 to 9 valid digits, depending on the application scenario. The number of valid digits does not depend on the decimal point.	32 bits	1.40129846432481707e-45 to 3.40282346638528860e+38, positive or negative	FLOAT

Usage:

- When a distributed query uses high-performance hardware instructions to perform single-precision or double-precision computing, the computing result may be slightly different because the execution sequence is different each time when an aggregate function, such as **SUM()** or **AVG()**, is invoked. Especially when the data volume is large (tens of millions or even billions of records), the computing result may be slightly different. In this case, you are advised to use the DECIMAL data type.
- An alias can be used to specify the data type.

  The following is an example:

  ```
  --Create a table that contains float data.
  CREATE TABLE float_t1 (float_col1 FLOAT) ;
  --Insert data of the float type.
  insert into float_t1 values (float '3.50282346638528862e+38');
  --View data.
  SELECT * FROM float_t1;
  float_col1
  ------------
  Infinity
  (1 row)
  --Drop the table.
  DROP TABLE float_t1;
  ```

- When the decimal part is 0, you can use **cast()** to convert the decimal part to an integer of the corresponding range. The decimal part is rounded off.

  The following is an example:


```
select CAST(1000.0001 as INT);
_col0
-------
1000
(1 row)
select CAST(122.5001 as TINYINT);
_col0
-------
123
(1 row)
```

- When an exponential expression is used, the string can be converted to the corresponding type.

  The following is an example:

```
select CAST(152e-3 as double);
_col0
-------
0.152
(1 row)
```

Character

Parameter	Description
VARCHAR(n)	Variable-length character string. *n* indicates the byte length.
CHAR(n)	Fixed-length character string. If the length is insufficient, spaces are added. *n* indicates the byte length. If the precision *n* is not specified, the default value **1** is used.
VARBINARY	Variable-length binary data. The value must be prefixed with **X**, for example, **X'65683F'**. Currently, a binary character string of a specified length is not supported.
JSON	The value can be **a JSON object**, **a JSON array**, **a JSON number**, **a JSON string**, **true**, **false** or **null**.
STRING	String compatible with impala. The bottom layer is varchar.
BINARY	Compatible with Hive BINARY. The underlying implementation is VARBINARY.

- In SQL expressions, simple character expressions and unicodes are supported. A unicode character string uses **U&** as the fixed prefix. An escape character must be added before Unicode that is represented by a 4-digit value.

```
-- Character expression
select 'hello,winter!';
      _col0
------------------
 hello,winter!
(1 row)
-- Unicode expression
select U&'Hello winter \2603 !';
      _col0
------------------
 Hello winter ☃ !
(1 row)
-- User-defined escape character
select U&'Hello winter #2603 !' UESCAPE '#';
      _col0
------------------
```



```
 Hello winter ⛄ !
(1 row)
```

- VARBINARY and BINARY

```
-- Create a VARBINARY or BINARY table.
 create table binary_tb(col1 BINARY);

--Insert data.
 INSERT INTO binary_tb values (X'63683F');

--Query data.
 select * from binary_tb ;  -- 63 68 3f
```

- When two CHARs with different numbers of spaces at the end are compared, the two CHARs are considered equal.

```
SELECT CAST('FO' AS CHAR(4)) = CAST('FO     ' AS CHAR(5));
 _col0
-------
 true
(1 row)
```

Time and Date Type

The time and date are accurate to milliseconds.

Table 11-75 Time and date type

Parameter	Description	Storage Space
DATE	Date and time. Only the ISO 8601 format is supported, for example **2020-01-01**.	32 bits
TIME	Time (hour, minute, second, millisecond) without a time zone Example: TIME '01:02:03.456'	64 bits
TIME WITH TIMEZONE	Time with a time zone (hour, minute, second, millisecond). Time zones are expressed as the numeric UTC offset value. Example: TIME '01:02:03.456 -08:00'	96 bits
TIMESTAMP	Timestamp	64 bits
TIMESTAMP WITH TIMEZONE	Timestamp with time zone	64 bits
INTERVAL YEAR TO MONTH	Time interval literal year and month, in the format of SY-M. S: optional symbols (+/-) Y: years M: months	128 bits

Parameter	Description	Storage Space
INTERVAL DAY TO SECOND	The time interval literally indicates the day, hour, minute, and second, and is accurate to millisecond. The format is SD H:M:S.nnn. S: optional symbols (+/-) D: days M: minutes S: seconds nnn: milliseconds	128 bits

The following is an example:

```
-- Query the date:
SELECT DATE '2020-07-08';
   _col0
------------
 2020-07-08
(1 row)

-- Query time:
SELECT TIME '23:10:15';
 _col0
--------------
 23:10:15
(1 row)

SELECT TIME '01:02:03.456 -08:00';
 _col0
--------------
 01:02:03.456-08:00
(1 row)

-- Time interval usage
SELECT TIMESTAMP '2015-10-18 23:00:15' + INTERVAL '3 12:15:4.111' DAY TO SECOND;
 _col0
-------------------------
 2015-10-22 11:15:19.111
(1 row)

SELECT TIMESTAMP '2015-10-18 23:00:15' + INTERVAL '3-1' YEAR TO MONTH;
          _col0
-------------------------
 2018-11-18 23:00:15
(1 row)

select INTERVAL '3' YEAR + INTERVAL '2' MONTH ;
 _col0
-------
 3-2
(1 row)

select INTERVAL '1' DAY+INTERVAL '2' HOUR +INTERVAL '3' MINUTE +INTERVAL '4' SECOND ;
     _col0
----------------
 1 02:03:04.000
(1 row)
```


ARRAY

Array

Example: **ARRAY[1, 2, 3]**.

```
-- Create an ARRAY type table.
create table array_tb(col1 ARRAY<STRING>);

-- Insert a record of the ARRAY type.
insert into array_tb values(ARRAY['HetuEngine','Hive','Mppdb']);

--Query data.
select * from array_tb; --     [HetuEngine, Hive, Mppdb]
```

MAP

Data type of a key-value pair.

Example: **MAP(ARRAY['foo', 'bar'], ARRAY[1, 2])**.

```
-- Create a Map table.
create table map_tb(col1 MAP<STRING,INT>);

-- Insert a piece of data of the Map type.
insert into map_tb values(MAP(ARRAY['foo','bar'],ARRAY[1,2]));

--Query data.
select * from map_tb; --    {bar=2, foo=1}
```

ROW

ROW fields can be any supported data type or a combination of different field data types.

```
-- Create a ROW table.
create table row_tb (id int,col1 row(a int,b varchar));

-- Insert data of the ROW type.
insert into row_tb values (1,ROW(1,'HetuEngine'));

--Query data.
select * from row_tb;
 id |     col1
----|--------------
  1 | {a=1, b=HetuEngine}

--Fields can be named. By default, a Row field is not named.
select  row(1,2e0),CAST(ROW(1, 2e0) AS ROW(x BIGINT, y DOUBLE));
           _col0           |     _col1
-------------------------|--------------
 {1, 2.0} | {x=1, y=2.0}
(1 row)

--Named fields can be accessed using the domain operator ".".
select col1.b from row_tb; -- HetuEngine

--Both named and unnamed fields can be accessed through location indexes, which start from 1 and must be a constant.
select col1[1] from row_tb;   -- 1
```

IPADDRESS

IP address, which can be an IPv4 or IPv6 address. In the system, however, this type of address is a unified IPv6 address.

IPv4 is supported by mapping IPv4 addresses to the IPv6 address range (RFC 4291#section-2.5.5.2). When an IPv4 address is created, it is mapped to an IPv6 address. During formatting, if the data is IPv4, the data will be mapped to IPv4 again. Other addresses are formatted according to the format defined in RFC 5952.

The following is an example:

```
select IPADDRESS '10.0.0.1', IPADDRESS '2001:db8::1';
  _col0   |    _col1
----------|-------------
 10.0.0.1 | 2001:db8::1
(1 row)
```

UUID

A standard universally unique identifier (UUID) is also called a globally unique identifier (GUID).

It complies with the format defined in RFC 4122.

The following is an example:

```
select UUID '12151fd2-7586-11e9-8f9e-2a86e4085a59';
                _col0
--------------------------------------
 12151fd2-7586-11e9-8f9e-2a86e4085a59
(1 row)
```

HYPERLOGLOG

Base statistics.

The cost of using HyperLogLog to approximate the count value of a unique number is far less than that of using count.

For details, see **HyperLogLog Functions**.

- HyperLogLog

  A HyperlogLog Sketch can be used to efficiently calculate the approximate value of **distinct()**.

  It begins with a sparse representation, then becomes a dense representation. And at this time efficiency becomes higher.

- P4HyperLogLog

  Similar to a HyperlogLog Sketch, but it starts with a dense representation.

QDIGEST

A quantile, also referred to as a quantile point, refers to that a probability distribution range of a random variable is divided into several equal numerical points. Commonly used quantile points include a median (that is, a 2-quantile), a 4-quartile, and a 100-percentile. Quantile digest is a set of quantiles. When the data to be queried is close to a quantile, the quantile can be used as the approximate value of the data to be queried. Its precision can be adjusted, but higher precision results in expensive space overhead.

STRUCT

The bottom layer is implemented using ROW. For details, see **ROW**.

The following is an example:

```
-- Creating a STRUCT table.
create table struct_tab (id int,col1 struct<col2: integer, col3: string>);

-- Insert data of the STRUCT type.
 insert into struct_tab VALUES(1, struct<2, 'HetuEngine'>);

--Query data.
select * from struct_tab;
 id |        col1
----|---------------------
  1 | {col2=2, col3=HetuEngine}
```

11.13.2 HetuEngine DDL SQL Syntax

11.13.2.1 CREATE SCHEMA

CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name

[COMMENT database_comment]

[LOCATION hdfs_path]

[WITH DBPROPERTIES (property_name=property_value,...)];

CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name

[WITH (property_name=property_value,...)]

Description

This statement is used to create an empty schema. A schema is a container for tables, views, and other database objects. If the optional parameter **IF NOT EXISTS** is specified and a schema with the same name already exists in the system, no error is reported.

The default schema path is **hdfs://hacluster/user/hive/warehouse/**.

Example

- Creating a schema named "web":
  ```
  CREATE SCHEMA web;
  ```
- Create a schema in the specified path. The following is an example:
  ```
  CREATE SCHEMA test_schema_5 LOCATION '/user/hive';
  ```
- Run the following command to create a schema named **sales** in the catalog named **Hive**:
  ```
  CREATE SCHEMA hive.sales;
  ```
- If the schema named **traffic** does not exist in the current catalog, run the following command to create a schema named **traffic**:
  ```
  CREATE SCHEMA IF NOT EXISTS traffic;
  ```
- Create a schema with attributes:
  ```
  CREATE DATABASE createtestwithlocation COMMENT 'Holds all values' LOCATION '/user/hive/warehouse/create_new' WITH dbproperties('name'='akku', 'id' ='9');
  ```



```
--Use the describe schema|database statement to query the schema created:
describe schema createtestwithlocation;
```

11.13.2.2 CREATE VIRTUAL SCHEMA

CREATE/DROP/SHOW VIRTUAL SCHEMA(S)

- **CREATE**

  The CREATE statement in HetuEngine is used to create a schema mapping and open the local domain data source to external systems based on the mapping information.

  The syntax is as follows:

  ```
  CREATE VIRTUAL SCHEMA [ IF NOT EXISTS ] [ ctlg_dest.]schema_name WITH ( [catalog =
  ctlg_name,] schema = schm_name, [property_name = expression, ...] )
  ```

  NOTE

  To create a virtual schema, provide the mapped schema information in **WITH**.

  ctlg_dest indicates the data source where the virtual schema is to be created. This parameter is optional. If this parameter is not specified, the catalog in the current session is used. If no catalogs are available in the current session, the creation will fail.

  WITH and **schema** are mandatory. The **catalog** parameter is optional. If catalog is not specified, the catalog in the current session is used.

  Example statement:

  ```
  CREATE VIRTUAL SCHEMA hive_default WITH (catalog = 'hive', schema = 'default');
  ```

- **DROP**

  The DROP statement in HetuEngine is used to delete a schema mapping.

  The syntax is as follows:

  ```
  DROP VIRTUAL SCHEMA [ IF EXISTS ] schema_name
  ```

  NOTE

  schema_name can be replaced with a fully qualified name (catalogName.virtualSchema).

  Example statement:

  ```
  DROP VIRTUAL SCHEMA hive_default;
  ```

- **SHOW**

  The SHOW statement in HetuEngine is used to query all schema mappings.

  The syntax is as follows:

  ```
  SHOW VIRTUAL SCHEMAS [ FROM catalog ] [ LIKE pattern ]
  ```

  Example statement:

  ```
  SHOW VIRTUAL SCHEMAS;
  ```

11.13.2.3 CREATE TABLE

Syntax

①

CREATE TABLE [IF NOT EXISTS]

[catalog_name.][db_name.]table_name (

{ column_name data_type [NOT NULL]

[COMMENT col_comment]

[WITH (property_name = expression [, ...])]

| LIKE existing_table_name

[{ INCLUDING | EXCLUDING } PROPERTIES]

}

[, ...]

)

[COMMENT table_comment]

[WITH (property_name = expression [, ...])]

②

CREATE [EXTERNAL] TABLE [IF NOT EXISTS]

[catalog_name.][db_name.]table_name (

{ column_name data_type [NOT NULL]

[COMMENT comment]

[WITH (property_name = expression [, ...])]

| LIKE existing_table_name

[{ INCLUDING | EXCLUDING } PROPERTIES]

}

[, ...]

)

[COMMENT 'table_comment']

[PARTITIONED BY(col_name data_type, ...)]

[CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name, col_name, ...)] INTO num_buckets BUCKETS]]

[ROW FORMAT row_format]

[STORED AS file_format]

[LOCATION 'hdfs_path']

[TBLPROPERTIES (orc_table_property = value [, ...])]

③

CREATE [EXTERNAL] TABLE [IF NOT EXISTS]

[catalog_name.][db_name.]table_name (

{ column_name data_type [NOT NULL]

[COMMENT comment]

[WITH (property_name = expression [, ...])]

| LIKE existing_table_name

[{ INCLUDING | EXCLUDING } PROPERTIES]

}

[, ...]

)

[PARTITIONED BY(col_name data_type, ...)]

[SORT BY ([*column* [, *column* ...]])]

[COMMENT 'table_comment']

[ROW FORMAT row_format]

[STORED AS file_format]

[LOCATION 'hdfs_path']

[TBLPROPERTIES (orc_table_property = value [, ...])]

Remarks

- **bucket_count** can be set for the **session** attribute. The default value is **-1**, indicating that **bucket_count** is not set. During partitioned table creation, if **bucket_count** is set to **-1** and **buckets** is not set in the table creation statement, the default value **16** is used.
- By default, external tables are stored in **/user/hive/warehouse/{*schema_name*}/{*table_name*}**. **schema_name** indicates the schema used for creating a table, and **table_name** indicates the table name.
- Setting "transactional=true" enables tables to support atomicity, consistency, isolation, and durability (ACID). However, after a table is defined as a transaction table, it cannot be degraded to a non-transaction table by setting "transactional=false".

  When transactional='true' or '0' is executed, type conversion will not be performed. Therefore, the following exception is thrown:

  Cannot convert ['true'] to boolean

  Cannot convert ['0'] to boolean
- By default, data cannot be inserted into a managed table whose property **external** is set to **true**. To use this function, add custom Hive property **hive.non-managed-table-writes-enabled=true**. For details, see **Precautions**.
- The MPPDB has a restriction that a database identifier can contain no more than 63 characters. If the identifier name exceeds the maximum length, the excess part is automatically truncated.
- Table creation is not supported in cross-domain scenarios.

Description

Creates a new empty table with specified columns by using CREATE TABLE. Use the **CREATE TABLE AS** statement to create a table with data.

- When the optional parameter **IF NOT EXISTS** is used, no error is reported if the table already exists.
- The WITH clause can be used to set properties for a newly created table or a single column, such as the storage location of the table and whether the table is an external table.
- The LIKE clause is used to include all column definitions from an existing table in a new table. You can specify multiple LIKE clauses to allow columns to be copied from multiple tables. If **INCLUDING PROPERTIES** is specified, all table properties are copied to the new table. If the attribute name specified in the WITH clause is the same as the copied attribute name, the value in the WITH clause is used. By default, the EXCLUDING PROPERTIES attribute is used. You can specify the INCLUDING PROPERTIES attribute for only one table.
- **PARTITIONED BY** can be used to specify the column of a partition. **CLUSTERED BY** can be used to specify columns for buckets. **SORT BY** and **SORTED BY** can be used to sort specified bucket columns. **BUCKETS** can be used to specify the number of buckets. **EXTERNAL** can be used to create foreign tables. **STORED AS** can be used to specify the file storage format. **LOCATION** can be used to specify the storage path in HDFS.

To view the supported column properties, run the following statement:

SELECT * FROM system.metadata.column_properties;

To view the supported table properties, run the following statement:

SELECT * FROM system.metadata.table_properties;

The following table lists the query result when the catalog is **hive**.

```
SELECT * FROM system.metadata.table_properties where catalog_name = 'hive';
```

catalog_name	property_name	default_value	type	description
hive	auto_purge	false	boolean	Skip trash when table or partition is deleted
hive	avro_schema_url	-	varchar	URI pointing to Avro schema for the table
hive	bucket_count	0	integer	Number of buckets
hive	bucketed_by	[]	array(varchar)	Bucketing columns
hive	bucketing_version	-	integer	Bucketing version
hive	csv_escape	-	varchar	CSV escape character
hive	csv_quote	-	varchar	CSV quote character

catalog_name	property_name	default_value	type	description
hive	csv_separator	-	varchar	CSV separator character
hive	external_location	-	varchar	File system location URI for external table
hive	format	ORC	varchar	Hive storage format for the table. Possible values: [ORC, PARQUET, AVRO, RCBINARY, RCTEXT, SEQUENCEFILE, JSON, TEXTFILE, TEXTFILE_MULTIDELIM, CSV]
hive	orc_compress	GZIP	varchar	Compression codec used. Possible values: [NONE, SNAPPY, LZ4, ZSTD, GZIP, ZLIB]
hive	orc_compress_size	262144	bigint	orc compression size
hive	orc_row_index_stride	10000	integer	no. of row index strides
hive	orc_stripe_size	67108864	bigint	orc stripe size
hive	orc_bloom_filter_columns	[]	array(varchar)	ORC Bloom filter index columns
hive	orc_bloom_filter_fpp	0.05	double	ORC Bloom filter false positive probability
hive	partitioned_by	[]	array(varchar)	Partition columns
hive	sorted_by	[]	array(varchar)	Bucket sorting columns
hive	textfile_skip_footer_line_count	-	integer	Number of footer lines
hive	textfile_skip_header_line_count	-	integer	Number of header lines
hive	transactional	false	boolean	Is transactional property enabled

Example

- Create a new table **orders** and use the WITH clause to specify the storage format, storage location, and whether the table is a foreign table.

The **auto.purge** parameter can be used to specify whether to clear related data when data removal operations (such as DROP, DELETE, INSERT OVERWRITE, and TRUNCATE TABLE) are performed.

- If it is set to **true**, metadata and data files are cleared.
- If it is set to **false**, only metadata is cleared and data files are moved to the HDFS trash bin. The default value is **false**. You are advised not to change the value. Otherwise, deleted data cannot be restored.

```
CREATE TABLE orders (
orderkey bigint,
orderstatus varchar,
totalprice double,
orderdate date
)
WITH (format = 'ORC', location='/user',orc_compress='ZLIB',external=true, "auto.purge"=false);

-- You can run the DESC FORMATTED statement to view details about table creation.
desc formatted  orders ;
                       Describe Formatted Table
-------------------------------------------------------------------------------
 # col_name      data_type      comment
 orderkey      bigint
 orderstatus      varchar
 totalprice      double
 orderdate      date

 # Detailed Table Information
 Database:                  default
 Owner:                     admintest
 LastAccessTime:            0
 Location:                  hdfs://hacluster/user
 Table Type:                EXTERNAL_TABLE

 # Table Parameters:
       EXTERNAL               TRUE
       auto.purge             false
       orc.compress.size      262144
       orc.compression.codec  ZLIB
       orc.row.index.stride   10000
       orc.stripe.size        67108864
       presto_query_id        20220812_084110_00050_srknk@default@HetuEngine
       presto_version         1.2.0-h0.cbu.mrs.320.r1-SNAPSHOT
       transient_lastDdlTime  1660293670

 # Storage Information
 SerDe Library:             org.apache.hadoop.hive.ql.io.orc.OrcSerde
 InputFormat:               org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
 OutputFormat:              org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
 Compressed:                No
 Num Buckets:               -1
 Bucket Columns:            []
 Sort Columns:              []
 Storage Desc Params:
       serialization.format   1
(1 row)
```

- Create a table with the specified row format.

```
-- When creating a table, use commas (,) to separate fields in each record in the data file.
CREATE TABLE student(
id string,birthday string,
grade int,
memo string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

-- When creating a table, specify the field delimiter and newline character as '\t' and '\n', respectively.
CREATE TABLE test(
id int,
name string ,
```



```
tel string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

- If the **orders** table does not exist, create the **orders** table and add table comments and column comments:

```
CREATE TABLE IF NOT EXISTS orders (
orderkey bigint,
orderstatus varchar,
totalprice double COMMENT 'Price in cents.',
orderdate date
)
COMMENT 'A table to keep track of orders.';
insert into orders values
(202011181113,'online',9527,date '2020-11-11'),
(202011181114,'online',666,date '2020-11-11'),
(202011181115,'online',443,date '2020-11-11'),
(202011181115,'offline',2896,date '2020-11-11');
```

- Create the **bigger_orders** table using the column definition of the **orders** table:

```
CREATE TABLE bigger_orders (
another_orderkey bigint,
LIKE orders,
another_orderdate date
);

SHOW CREATE TABLE bigger_orders ;
                        Create Table
-----------------------------------------------------------------------
 CREATE TABLE hive.default.bigger_orders (
    another_orderkey bigint,
    orderkey bigint,
    orderstatus varchar,
    totalprice double,
    ordersdate date,
    another_orderdate date
 )
 WITH (
    external = false,
    format = 'ORC',
    location = 'hdfs://hacluster/user/hive/warehouse/bigger_orders',
    orc_compress = 'GZIP',
    orc_compress_size = 262144,
    orc_row_index_stride = 10000,
    orc_stripe_size = 67108864
 )
(1 row)
```

- **①Example of creating a table:**

```
CREATE EXTERNAL TABLE hetu_test (orderkey bigint, orderstatus varchar, totalprice double, orderdate date) PARTITIONED BY(ds int) SORT BY (orderkey, orderstatus) COMMENT 'test' STORED AS ORC LOCATION '/user' TBLPROPERTIES (orc_compress = 'SNAPPY', orc_compress_size = 6710422, orc_bloom_filter_columns = 'orderstatus,totalprice');
```

- **②Example of creating a table:**

```
CREATE EXTERNAL TABLE hetu_test1 (orderkey bigint, orderstatus varchar, totalprice double, orderdate date) COMMENT 'test' PARTITIONED BY(ds int) CLUSTERED BY (orderkey, orderstatus) SORTED BY (orderkey, orderstatus) INTO 16 BUCKETS STORED AS ORC LOCATION '/user' TBLPROPERTIES (orc_compress = 'SNAPPY', orc_compress_size = 6710422, orc_bloom_filter_columns = 'orderstatus,totalprice');
```

- **③Example of creating a table:**

```
CREATE TABLE hetu_test2 (orderkey bigint, orderstatus varchar, totalprice double, orderdate date, ds int) COMMENT 'This table is in Hetu syntax' WITH (partitioned_by = ARRAY['ds'], bucketed_by = ARRAY['orderkey', 'orderstatus'], sorted_by = ARRAY['orderkey', 'orderstatus'], bucket_count = 16, orc_compress = 'SNAPPY', orc_compress_size = 6710422, orc_bloom_filter_columns = ARRAY['orderstatus', 'totalprice'], external = true, format = 'orc', location = '/user');
```


- Run the following statements to view the table:

```
show create table hetu_test1;
                        Create Table
-----------------------------------------------------------------------
 CREATE TABLE hive.default.hetu_test1 (
    orderkey bigint,
    orderstatus varchar,
    totalprice double,
    orderdate date,
    ds integer
 )
 COMMENT 'test'
 WITH (
    bucket_count = 16,
    bucketed_by = ARRAY['orderkey','orderstatus'],
    bucketing_version = 1,
    external_location = 'hdfs://hacluster/user',
    format = 'ORC',
    orc_bloom_filter_columns = ARRAY['orderstatus','totalprice'],
    orc_bloom_filter_fpp = 5E-2,
    orc_compress = 'SNAPPY',
    orc_compress_size = 6710422,
    orc_row_index_stride = 10000,
    orc_stripe_size = 67108864,
    partitioned_by = ARRAY['ds'],
    sorted_by = ARRAY['orderkey','orderstatus']
 )
(1 row)
```

Create a partitioned table.

```
--Create a schema.
CREATE SCHEMA hive.web WITH (location = 'hdfs://hacluster/user');
--Create a partitioned table.
CREATE TABLE hive.web.page_views (
  view_time timestamp,
  user_id bigint,
  page_url varchar,
  ds date,
  country varchar
)
WITH (
  format = 'ORC',
  partitioned_by = ARRAY['ds', 'country'],
  bucketed_by = ARRAY['user_id'],
  bucket_count = 50
);
--Insert an empty partition.
CALL system.create_empty_partition(
    schema_name => 'web',
    table_name => 'page_views',
    partition_columns => ARRAY['ds', 'country'],
    partition_values => ARRAY['2020-07-17', 'US']);

CALL system.create_empty_partition(
    schema_name => 'web',
    table_name => 'page_views',
    partition_columns => ARRAY['ds', 'country'],
    partition_values => ARRAY['2020-07-18', 'US']);

--View the partition.
SELECT * FROM hive.web."page_views$partitions";
     ds     | country
------------|---------
 2020-07-18 | US
 2020-07-17 | US
--Insert data.
insert into hive.web.page_views values(timestamp '2020-07-17 23:00:15',bigint '15141','www.local.com',date
'2020-07-17','US' );
```



```
insert into hive.web.page_views values(timestamp '2020-07-18 23:00:15',bigint '18148','www.local.com',date
'2020-07-18','US' );

--Query data.
 select * from hive.web.page_views;
        view_time         | user_id |   page_url    |     ds     | country
--------------------------|---------|---------------|------------|---------
 2020-07-17 23:00:15.000 |   15141 | www.local.com | 2020-07-17 | US
2020-07-18 23:00:15.000 |   18148 | www.local.com | 2020-07-18 | US
```

11.13.2.4 CREATE TABLE AS

Syntax

CREATE [EXTERNAL]① TABLE [IF NOT EXISTS] [catalog_name.] [db_name.]table_name [(column_alias, ...)]

[[PARTITIONED BY ①(col_name, ...)] [SORT BY① ([*column* [, *column* ...]])]]①

[COMMENT 'table_comment']

[WITH (property_name = expression [, ...])]②

[[STORED AS file_format]①

[LOCATION 'hdfs_path']①

[TBLPROPERTIES (orc_table_property = value [, ...])]]①

AS query

[WITH [NO] DATA]②

Remarks

The syntax of ① and ② cannot be used together.

When the **avro_schema_url** attribute is used:

- **CREATE TABLE AS** is not supported.
- When **CREATE TABLE** is used, **partitioned_by** and **bucketed_by** are not supported.
- Columns cannot be modified by using **alter table**.

Description

It is used to create a table that contains the **SELECT** query result.

Use the **CREATE TABLE** statement to create an empty table.

When the **IF NOT EXISTS** clause is used, no error is reported if the table already exists.

The **WITH** clause can be used to set the properties of a newly created table, such as the storage location of the table and whether the table is an external table.

Example

- Run the following statement to create the **orders_column_aliased** table based on the query result of a specified column:
```
CREATE TABLE orders_column_aliased (order_date, total_price)
AS
SELECT orderdate, totalprice FROM orders;
```
- Create the **orders_by_data** table based on the summary result of the **orders** table.
```
CREATE TABLE orders_by_date
COMMENT 'Summary of orders by date'
WITH (format = 'ORC')
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
```
- If the **orders_by_date** table does not exist, create the **orders_by_date** table:
```
CREATE TABLE IF NOT EXISTS orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
```
- Create the **empty_orders table** using the schema that is the same as that of the **orders** table but does not contain data:
```
CREATE TABLE empty_orders AS
SELECT *
FROM orders
WITH NO DATA;
```
- Use **VALUES** to create a table. For details, see **VALUES**.
- Partitioned table example:
```
CREATE EXTERNAL TABLE hetu_copy(corderkey, corderstatus, ctotalprice, corderdate, cds)
 PARTITIONED BY(cds)
 SORT BY (corderkey, corderstatus)
 COMMENT 'test'
 STORED AS orc
 LOCATION '/user/hetuserver/tmp'
 TBLPROPERTIES (orc_bloom_filter_fpp = 0.3, orc_compress = 'SNAPPY', orc_compress_size = 6710422,
orc_bloom_filter_columns = 'corderstatus,ctotalprice')
 as select * from hetu_test;

 CREATE TABLE hetu_copy1(corderkey, corderstatus, ctotalprice, corderdate, cds)
 WITH (partitioned_by = ARRAY['cds'], bucketed_by = ARRAY['corderkey', 'corderstatus'],
 sorted_by = ARRAY['corderkey', 'corderstatus'],
 bucket_count = 16,
 orc_compress = 'SNAPPY',
 orc_compress_size = 6710422,
 orc_bloom_filter_columns = ARRAY['corderstatus', 'ctotalprice'],
 external = true,
 format = 'orc',
 location = '/user/hetuserver/tmp ')
  as select * from hetu_test;
```

11.13.2.5 CREATE TABLE LIKE

Syntax

CREATE TABLE [IF NOT EXISTS] table_name ({coulumn_name data_type [COMMENT comment] [WITH (property_name = expression [, ...])] | LIKE existing_table_name [{INCLUDING| EXCLUDING} PROPERTIES]}) [, ...] [COMMENT table_comment] [WITH (property_name = expression [, ...])]

Description

The **LIKE** clause allows you to include all column definitions of an existing table in a new table. You can use multiple **LIKE** statements to copy the columns of multiple tables.

If the **INCLUDING PROPERTIES** option is used, all attributes of the table are copied to the new table. This option takes effect for only one table.

You can use the **WITH** clause to modify the name of the attribute copied from the table.

The **EXCLUDING PROPERTIES** attribute is used by default.

For a table with partitions, if the **LIKE** clause is enclosed in parentheses, the copied column definition does not contain information about the partition key.

Example

- Create basic tables order01 and order02.
  ```
  CREATE TABLE order01(id int,name string,tel string) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'STORED AS TEXTFILE;
  CREATE TABLE order02(sku int, sku_name string, sku_describe string);
  ```
- Create the orders_like01 table that contains the columns defined by the order01 table and the table properties.
  ```
  CREATE TABLE orders_like01 like order01 INCLUDING PROPERTIES;
  ```
- Create the orders_like02 table that contains the columns defined by the order02 table, and set the storage format of the table to **TEXTFILE**.
  ```
  CREATE TABLE orders_like02 like order02 STORED AS TEXTFILE;
  ```
- Create the orders_like03 table that contains the columns defined by the order01 table and the table properties, the columns defined by the order02 table, and additional columns c1 and c2.
  ```
  CREATE TABLE orders_like03 (c1 int,c2 float,LIKE order01 INCLUDING PROPERTIES,LIKE order02);
  ```
- Create the orders_like04 and orders_like05 tables. Both tables contain the definition of the same order_partition. The orders_like04 table does not contain but the orders_like05 table contains the partition key information.
  ```
  CREATE TABLE order_partition(id int,name string,tel string) PARTITIONED BY (sku int);
  CREATE TABLE orders_like04 (like order_partition);
  CREATE TABLE orders_like05 like order_partition;
  DESC orders_like04;
   Column |  Type   | Extra | Comment
  --------|---------|-------|---------
   id     | integer |       |
   name   | varchar |       |
   tel    | varchar |       |
   sku    | integer |       |
  (4 rows)

  DESC orders_like05;

   Column |  Type   |     Extra     | Comment
  --------|---------|---------------|---------
   id     | integer |               |
   name   | varchar |               |
   tel    | varchar |               |
   sku    | integer | partition key |
  (4 rows)
  ```


11.13.2.6 CREATE VIEW

Syntax

CREATE [OR REPLACE] VIEW view_name [(column_name [COMMENT 'column_comment'][, ...])] [COMMENT 'view_comment'] [TBLPROPERTIES (property_name = property_value)] AS query

Remarks

Only Catalog of the Hive data source supports the column description of the view.

For the views created in HetuEngine, the definition of the views is stored in the data source in encoding mode. You can query the view in the data source but cannot perform operations on the view.

Views are read-only. You cannot perform the LOAD or INSERT operation on views.

A view can contain the **ORDER BY** and **LIMIT** clauses. If a query statement associated with the view also contains these clauses, the **ORDER BY** and **LIMIT** clauses in the query statement are calculated based on the result of the view.

Description

It is used to create a view using the SELECT query result. A view is a logical table that can be referenced by future queries. A view has no data. The query corresponding to the view is executed each time the view is referenced by another query.

If the view already exists, the optional ORREPLACE clause causes the view to be replaced without reporting an error.

Example

- To create a view named **test** in the **orders** table:
```
CREATE VIEW test (oderkey comment 'orderId',orderstatus comment 'status',half comment 'half') AS
SELECT orderkey, orderstatus, totalprice / 2 AS half FROM orders;
```
- Create the **orders_by_date** view based on the summary result of the **orders** table.
```
CREATE VIEW orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
```
- Create a view to replace the existing view:
```
CREATE OR REPLACE VIEW test AS
SELECT orderkey, orderstatus, totalprice / 4 AS quarter
FROM orders
```
- Create a view and set table attributes:
```
create or replace view  view1 comment 'the first view' TBLPROPERTIES('format'='orc') as select * from fruit;
```

Precautions

When **alter** is used to modify the table on which the created view depends, you need to create the view again. Otherwise, an error will be reported when you query the view again.

11.13.2.7 CREATE FUNCTION

Syntax

```
CREATE FUNCTION qualified_function_name (
parameter_name parameter_type
[, ...]
)
RETURNS return_type
[ COMMENT function_description ]
[ LANGUAGE [ JAVA ] ]
[ SPECIFIC specificName ]
[ DETERMINISTIC | NOT DETERMINISTIC ]
[ RETURNS NULL ON NULL INPUT | CALLED ON NULL INPUT ]
[ SYMBOL class_name ]
[ URI hdfs_path_to_jar ]
```

Description

It is used to create a function based on a given definition.

- Each function is uniquely identified by a qualified name and parameter type list. The value format of **qualified_function_name** must be *catalog.schema.function_name*. You can plan and manage the function namespace (its format is *catalog.schema*). It is irrelevant to the concepts of catalog and schema in HetuEngine. The value of **parameter_type** must be a data type supported by HetuEngine.
- The value of **return_type** must be a data type supported by HetuEngine and must match the actual type returned by the function. Forcible type conversion is not executed.
- You can specify a group of features to decorate a function and specify its behavior. Each feature can be specified only once. For details, see **Table 11-76**.

Table 11-76 Feature description

Feature	Default Value	Description
Language clause	-	Language used to define a function. Currently, Java is supported. • Java function: A JAR file for implementing a function needs to be provided, and the JAR file needs to be placed in HDFS that can be read by HetuEngine.

Feature	Default Value	Description
Deterministic characteristic	NOT DETERMINISTIC	Whether a function is deterministic. • **DETERMINISTIC**: A function is considered deterministic if it always returns the same result set when called with the same input set. • **NOT DETERMINISTIC**: A function is considered nondeterministic if it does not return the same result set when called with the same input set.
Null-call clause	CALLED ON NULL INPUT	Function behavior. • **RETURNS NULL ON NULL INPUT**: When **NULL** is used as a function parameter, **NULL** is returned. • **CALLED ON NULL INPUT**: When **NULL** is used as a function parameter, the function is called.
Symbol class_name	-	Used by a Java function to specify a fully qualified class name for function implementation.
Uri hdfs_path_to_jar	-	Used by a Java function to specify a JAR file path for function implementation.

Remarks

- Permissions are controlled only based on user groups. For details, see **Table 11-77**.

Table 11-77 Permission control

Operation	Permission Control
CREATE	No permission control
DROP	Only the owner has the permission to perform the operation.
SELECT	No permission control
SHOW	No permission control

Example

- Create a Java function **example.default.add_two**. (You need to build and deploy the UDF first.)


```
CREATE FUNCTION example.default.add_two (
 num integer
)
RETURNS integer
LANGUAGE JAVA
DETERMINISTIC
SYMBOL "com.example.functions.AddTwo"
URI "hdfs://hacluster/udfs/function-1.0.jar";

-- Execute the function.
select hetu.default.add_two(2);
```

11.13.2.8 CREATE MATERIALIZED VIEW

Syntax

CREATE MATERIALIZED VIEW [IF NOT EXISTS] view_name [COMMENT string] [WITH properties] AS query

Description

This statement is used to create a materialized view based on the SELECT query result. A materialized view is a database object that contains the results of a query. For example, it can be a local copy of remote data, a row or column, or a subset of rows and columns in the query results of a single table or multiple joined tables, or a summary table that uses aggregate functions.

The materialized view is usually created using precomputed aggregation and join results of a query. Materialized views support the query rewriting feature, which transforms a query statement based on a base table to an equivalent based on one or more materialized views.

The syntax supports the following attributes:

- **storage_table**: specifies the name of a storage table.
- **need_auto_refresh**: During compute instance management, after creating a maintenance instance, you can set **need_auto_refresh** to **true** to create a materialized view that can be automatically refreshed as well as create and submit an automatic refresh task for the created materialized view. On this basis, you can set **refresh_duration**, **start_refresh_ahead_of_expiry**, **refresh_priority**, and other properties to adjust the automatic refresh task.
- **mv_validity**: specifies the life cycle of a materialized view. **0** indicates that the materialized view is permanently valid. The minimum value is 1 minute. If **need_auto_refresh** is set to **false**, **mv_validity** is set to **0** by default. If **need_auto_refresh** is set to **true**, **mv_validity** is set to 24 hours by default.
- **refresh_duration**: specifies the maximum waiting duration of the automatic materialized view refresh task. The default value is 5 minutes. The value ranges from 1 minute to 24 hours. If the waiting time of the automatic refresh task exceeds the maximum waiting time, the task status is displayed as **timeout** on the automatic task page.
- **start_refresh_ahead_of_expiry**: specifies the submission time of automatic materialized refreshing based on **mv _validity**. This attribute indicates that automatic refreshing tasks are submitted when the specified percentage of the materialized view life cycle is reached. The default value is **0.2**, and the minimum value is **0.05**.

- **refresh_priority**: priority for automatic refreshing tasks of the materialized views. The default and maximum value is **3**. **1** indicates the highest priority. Tasks with higher priorities are more likely to be executed first.

Example

- Create the same schema in **mv catalog** and the catalog for data storage (the following example uses Hive as the catalog for data storage), and enable query rewrite for the materialized view.
```
hetuengine:tpcds_2gb> set session materialized_view_rewrite_enabled=true;
hetuengine:tpcds_2gb> create schema mv.tpcds;
CREATE SCHEMA
hetuengine:tpcds_2gb> create schema hive.tpcds;
CREATE SCHEMA
```
- Create a table.
```
hetuengine:tpcds_2gb> create table t1 (id int, c1 varchar);
hetuengine:tpcds_2gb> Insert into t1 values (1,'abc'), (2,'abc2'), (3,'abc3'), (4,'abc4'), (5,'abc5'),(6,
'abc6');
hetuengine:tpcds_2gb> create table tb_a(a int ,b varchar, c varchar);
hetuengine:tpcds_2gb> create table tb_b(a int ,d varchar, e varchar);
```
- Create a materialized view named **mv.tpcds.test** in **tpcds schema** of **mv catalog**. If a materialized view with the same name already exists, an error occurs.
```
hetuengine:tpcds_2gb> create materialized view mv.tpcds.test as select c1 from t1 where id <7;
CREATE MATERIALIZED VIEW
```
- Create a materialized view named **mv.tpcds.test** based on a specified column name in **mv catalog** and **tpcds schema**.
```
hetuengine:tpcds_2gb> create materialized view mv.tpcds.test (a ,b) as select c1, id from t1 where
id<7;
CREATE MATERIALIZED VIEW
```
- Create a materialized view using **if not exists** in **mv catalog** and **tpcds schema**. If a materialized view with the same name already exists, no error occurs.
```
hetuengine:tpcds_2gb> create materialized view if not exists mv.tpcds.test as select c1, id from t1
where id<7;
CREATE MATERIALIZED VIEW
```
- Create a materialized view with specified properties in **mv catalog** and **tpcds schema**.
```
hetuengine:tpcds_2gb> create materialized view mv.tpcds.test with
(storage_table='mppdb.tpcds.test2',need_auto_refresh = true, mv_validity = '10m',
start_refresh_ahead_of_expiry = 0.2, refresh_priority = 1, refresh_duration = '5m') as select c1, id from
t1 where id<7;
CREATE MATERIALIZED VIEW
```
- Create a materialized view with comments.
```
hetuengine:tpcds_2gb> create materialized view mv.tpcds.test comment 'test_comment' as select
c1, id from t1 where id<7;
CREATE MATERIALIZED VIEW
```

Precautions

- To create a materialized view, ensure that **mv catalog** exists.
- After creating a materialized view, you need to run the **refresh materialized view** *xxx* command to populate the materialized view with data.
- The materialized view rewriting function needs to be enabled at the system or session level.
- Schema used to create a view in **mv catalog** must be created in advance in the **catalog** and **mv catalog** that are used for data storage.

- Do not delete the data table of a materialized view in the catalog.
- Do not create a materialized view using the query statement that contains **Order By**.
- When creating an MV, ensure that the query does not contain subqueries or subquery joins. Otherwise, use WITH subqueries instead.

  Example:

  ```
  hetuengine:tpcds_2gb> create materialized view mv.tpcds.test1 as select t1.a, b, d from ((select a, b, c from tb_a) as t1 join (select a, d, e from tb_b) as t2 on t1.a=t2.a);
  ```

  Use WITH to replace subqueries and subquery joins:

  ```
  hetuengine:tpcds_2gb>
  create materialized view mv.tpcds.test1 as
  with t1 as (select a, b, c from tb_a),
  t2 as (select a, d, e from tb_b)select t1.a, b, d from t1 join t2 on t1.a = t2.a;
  ```

- Querying the rewrite of some materialized views is not supported. If the data to be queried is part of data (subset data) of the view, the query statement cannot be rewritten to an equivalent one for querying the materialized view.

  For example, if you use the **select id from test where id <100** statement to create a materialized view named **t1** and then perform a query the using the **select id from test where id <50** statement, the query statement cannot be rewritten because it attempts to use some data of the materialized view.
- When a materialized view is created, the table name must be a fully qualified name (catalogName.schemaName.tableName) or a table name.

  Example:

  ```
  hetuengine:tpcds_2gb> create materialized view mv.tpcds.test as select c1 from t1 where id <7;
  ```

  Table name **t1** can be replaced with fully qualified name **hive.tpcds_2gb.t1** but cannot be replaced with **tpcds_2gb.t1**.
- Query rewrite for materialized views does not support full table scan. SQL queries do not use the Where clause and cannot be rewritten by queries.

  For example, if the column definition of the **hivetb1** table contains the **id**, **name**, and **age** columns, the following SQL query cannot be rewritten:

  ```
  Create MV SQL : select id,name,age from hivetb1;
  ```

11.13.2.9 ALTER MATERIALIZED VIEW STATUS

Syntax

```
ALTER MATERIALIZED VIEW qualifiedName SET STATUS <status>
```

Description

To change the status of a materialized view, you can only convert between **ENABLE** and **SUSPEND**. The **DISABLE** state can be changed to **SUSPEND** or **ENABLE** only. The statuses of materialized views include:

- **INIT**: The materialized view is created for the first time.
- **SUSPEND**: The materialized view is suspended. The suspended materialized view is not rewritten.
- **ENABLE**: The materialized view is available.
- **REFRESHING**: The materialized view data is being refreshed and cannot be rewritten.

- **DISABLE**: The materialized view is disabled.

Example

Change the status of **mv.default.mv1** to **SUSPEND**.

```
alter materialized view mv.default.mv1 set status SUSPEND;
```

11.13.2.10 ALTER MATERIALIZED VIEW

Syntax

```
ALTER MATERIALIZED VIEW QUALIFIEDNAME SET PROPERTIES
PROPERTY_NAME=PROPERTY_VALUE;
```

Description

Modify the attributes of materialized views. For details about the attributes, see **CREATE MATERIALIZED VIEW**.

Example

Change the PROPERTIES attribute of the automatic refresh task submitted by the materialized view of mv.default.mv1mv.mvtestprop.pepa_ss to refresh_priority = 2.

```
Alter materialized view mv.mvtestprop.pepa_ss set PROPERTIES refresh_priority = 2;
```

11.13.2.11 ALTER TABLE

Syntax

NOTE

name, **new_name**, **column_name**, **new_column_name**, and **table_name_*** are user-defined parameters.

1. The following statement is used to rename a table.

   ALTER TABLE name RENAME TO new_name

2. Change the column name of the table and add comments (optional) and properties (optional) to the column. For details about supported column properties, see **Description**.

   ALTER TABLE name ADD COLUMN column_name data_type [COMMENT comment] [WITH (property_name = expression [, ...])]

3. The following statement is used to delete the **column_name** column from the table.

   ALTER TABLE name DROP COLUMN column_name

NOTICE

- Partition or bucket columns cannot be deleted.
- DROP COLUMN does not support tables stored in RCTEXT, RCBINARY, or RCFILE format. Connector accesses columns in different file formats in different modes. The query may fail after the **DROP COLUMN** operation is performed. For example:
  - For a non-partitioned table stored in ORC format, if the query fails after the **DROP COLUMN** operation is performed, run the following command to set the Session property:

    set session hive.orc_use_column_names=true;
  - For a non-partitioned table stored in Parquet format, if the query fails after the **DROP COLUMN** operation is performed, run the following command to set the Session property:

    set session hive.parquet_use_column_names=true;
  - For partitioned or transaction tables in ORC or Parquet format, session properties cannot be configured to ensure query success after the **DROP COLUMN** operation is performed.

4. The following statement is used to rename the **column_name** column to **new_column_name**.

   ALTER TABLE name RENAME COLUMN column_name TO new_column_name

   NOTICE

   Partition or bucket columns cannot be renamed.

5. The following statement is used to add partitions to a partitioned table.

   ALTER TABLE name ADD [IF NOT EXISTS] PARTITION partition_spec [LOCATION 'location'][PARTITION partition_spec [LOCATION 'location'], ...];

6. The following statement is used to delete a partition from a partitioned table. This operation deletes data and metadata from the partition. If the directory for storing partition is specified when **ADD PARTITION** is run, the folder where the partition is located and data will not be deleted after **DROP PARTITION** is run, regardless of whether the table is an internal table or external table. If the directory for storing partition is specified when **ADD PARTITION** is run, the directory will be deleted from HDFS and data is moved to the **.Trash/Current** folder.

   ALTER TABLE table_name DROP [IF EXISTS] PARTITION partition_spec[, PARTITION partition_spec, ...];

   NOTICE

   When an external Hive data source is used, if the partition key is a fixed-length string, for example, char(5), and the string length of the corresponding data is fewer than five characters, the **drop partition** operation fails.

7. The following statement is used to rename a partition.

   ALTER TABLE table_name PARTITION(partition_key = partition_value1) rename to partition(partition_key = partition_value2)

8. The following statement is used to migrate the partition of **table_name_1** to **table_name_2**.

   ALTER TABLE table_name_2 EXCHANGE PARTITION (partition_spec) WITH TABLE table_name_1;

9. The following statement is used to migrate multiple partitions of **table_name_1** to **table_name_2**.

   ALTER TABLE table_name_2 EXCHANGE PARTITION (partition_spec, partition_spec2, ...) WITH TABLE table_name_1;

10. The following statements are used to add or modify table properties.

    ALTER TABLE table_name SET TBLPROPERTIES (property_name = property_value[, property_name = property_value, ...]);

    NOTE

    TBLPROPERTIES allows you to add or modify table attributes supported by a connector in key-value pair mode (attribute names and attributes must be strings enclosed in single or double quotation marks). The following uses the Hive connector as an example:

    - TBLPROPERTIES ("transactional"="true"). The value can be **true** or **false**.
    - TBLPROPERTIES ("auto.purge"="true"). The value can be **true** or **false**.

11. The following statement is used to modify column properties.

    ALTER TABLE table_name [PARTITION partition_spec] CHANGE [COLUMN] col_old_name col_new_name column_type [COMMENT col_comment] [FIRST|AFTER column_name] [CASCADE|RESTRICT]

    NOTICE

    - For an existing table, modify the column name, data type, comment, location (**[FIRST|AFTER column_name]** is used to specify the location of the column after modification), or any combination of the preceding items. If a partition clause is included in the syntax, the metadata of the corresponding partition also changes. In **CASCADE** mode, the syntax will take effect on the metadata of the table and table partition. In the default **RESTRICT** mode, the modification to a column takes effect only on the metadata of the table.
    - The column modification statement can modify only the metadata of a table or partition, but cannot modify the data itself. Ensure that the actual data layout of the table or partition complies with the metadata definition.
    - The partition column or bucket column of a table and Optimized Row Columnar (ORC) tables cannot be modified.

12. The following statement is used to change the storage location of the table or partition.

    ALTER TABLE table_name [PARTITION partition_spec] SET LOCATION location;

NOTE

- You can run the **ALTER TABLE [PARTITION] SET** statement to set the table or partition location.
- After the **SET LOCATION** statement is run, table or partition data may not be displayed.
- When a table or partition directory is created, **SET LOCATION** uses the specified directory instead of the default directory created on Hive during the creation of the table or partition.
- This statement does not affect the original data in the table or partition, or modify the original table or partition directory. New data is saved to the new directory.

13. The following statement is used to change the format of the data file of a table or partition.

    ALTER TABLE table_name [PARTITION partition_spec] SET FILEFORMAT file_format;

    NOTE

    - This operation changes the metadata of a table or partition and the type of the inventory data file. The operation cannot be performed at the SQL layer and can only be performed externally.
    - The following file formats are supported: AVRO, PARQUET, ORC, RCFILE, TEXTFILE, and SEQUENCEFILE.

14. The following statement is used to modify the physical storage properties of a table.

    ALTER TABLE table_name CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name, ...)] INTO num_buckets BUCKETS;

Remarks

- EXCHANGE PARTITION:
  - The single or multiple partitions to be migrated must exist and belong to the source table before migration, and the partitions are not included in the target table.
  - Tables involved in this operation must have the same column definition and partition key.
  - If the table contains indexes, the operation fails.
  - If either the source table or the target table is a transaction table, the **EXCHANGE PARTITION** operation is not allowed.
  - The operation result for the target table is that multiple partitions are successfully migrated at the same time or fail to be migrated. For the source table, all migrated partitions are released after the operation is successful.
  - The **ALTER TABLE** statement for column modification does not support ORC tables.
- The **ALTER TABLE table_name ADD | DROP col_name** statement is available only for non-partitioned tables in ORC or PARQUET format.

Example

- To change the table name from **users** to **people**:

  ALTER TABLE *users* **RENAME TO** *people*;

- To add the **zip** column to the **users** table:

  ALTER TABLE *users* **ADD COLUMN** *zip* **varchar;**
- To delete the **zip** column from the **users** table:

  ALTER TABLE *users* **DROP COLUMN** *zip*;
- To change the column name **id** in the **users** table to **user_id**:

  ALTER TABLE *users* **RENAME COLUMN id TO** *user_id*;
- To modify a partition:

```
--Create two partitioned tables.
CREATE TABLE IF NOT EXISTS hetu_int_table5 (eid int, name String, salary String, destination String, dept String, yoj int) COMMENT 'Employee Names' partitioned by (dt timestamp,country String, year int, bonus decimal(10,3)) STORED AS TEXTFILE;

CREATE TABLE IF NOT EXISTS hetu_int_table6 (eid int, name String, salary String, destination String, dept String, yoj int) COMMENT 'Employee Names' partitioned by (dt timestamp,country String, year int, bonus decimal(10,3)) STORED AS TEXTFILE;

--Add partitions.
ALTER TABLE hetu_int_table5 ADD IF NOT EXISTS PARTITION (dt='2008-08-08 10:20:30.0', country='IN', year=2001, bonus=500.23) PARTITION (dt='2008-08-09 10:20:30.0', country='IN', year=2001, bonus=100.50) ;

--View the partition.
show partitions hetu_int_table5;
           dt            | country | year |  bonus
-------------------------|---------|------|---------
 2008-08-09 10:20:30.000 | IN      | 2001 | 100.500
 2008-08-08 10:20:30.000 | IN      | 2001 | 500.230
(2 rows)

--Delete a partition.
ALTER TABLE hetu_int_table5 DROP IF EXISTS PARTITION (dt=timestamp '2008-08-08 10:20:30.0', country='IN', year=2001, bonus=500.23);

--View the partition.
show partitions hetu_int_table5;
           dt            | country | year |  bonus
-------------------------|---------|------|---------
 2008-08-09 10:20:30.000 | IN      | 2001 | 100.500
(1 row)

--Example of migrating a partition
CREATE SCHEMA part_test;
CREATE TABLE hetu_exchange_partition1 (a string, b string) PARTITIONED BY (ds string);
CREATE TABLE part_test.hetu_exchange_partition2 (a string, b string) PARTITIONED BY (ds string);
ALTER TABLE hetu_exchange_partition1 ADD PARTITION (ds='1');

--View the partition.
 show partitions hetu_exchange_partition1;
 ds
----
 1
(1 row)

show partitions part_test.hetu_exchange_partition2;
 ds
----
(0 rows)

--Migrate the partition from table 1 to table 2.
ALTER TABLE part_test.hetu_exchange_partition2 EXCHANGE PARTITION (ds='1') WITH TABLE hetu_exchange_partition1;

--View the partition again. The partition is successfully migrated.
show partitions hetu_exchange_partition1;
 ds
```



```
----
 (0 row)

show partitions part_test.hetu_exchange_partition2;
 ds
----
1
(1 rows)

--Rename a partition.
CREATE TABLE IF NOT EXISTS hetu_rename_table ( eid int, name String, salary String, destination
String, dept String, yoj int)
COMMENT 'Employee details'
partitioned by (year int)
STORED AS TEXTFILE;

ALTER TABLE hetu_rename_table ADD IF NOT EXISTS PARTITION (year=2001);

SHOW PARTITIONS hetu_rename_table;
year
------
 2001
(1 row)

ALTER TABLE hetu_rename_table PARTITION (year=2001) rename to partition (year=2020);

SHOW PARTITIONS hetu_rename_table;
year
------
 2020
(1 row)

--Modify a partitioned table.
create table altercolumn4(a integer, b string) partitioned by (c integer);

--Modify the file format of the table.
alter table altercolumn4 SET FILEFORMAT textfile;

insert into altercolumn4 values (100, 'Daya', 500);

alter table altercolumn4 partition (c=500) change column b empname string comment 'changed
column name to empname' first;

--Change the storage location of the partitioned table. (You need to create a directory in HDFS. After
the statement is run, the inserted data cannot be queried.)
alter table altercolumn4 partition (c=500) set Location '/user/hive/warehouse/c500';

--Change the name of column b to name and the data type from integer to string.
create table altercolumn1(a integer, b integer) stored as textfile;

alter table altercolumn1 change column b name string;

--Modify the storage property of altercolumn1.
ALTER TABLE altercolumn1 CLUSTERED BY(a, name) SORTED BY(name) INTO 25 BUCKETS;

--View the properties of altercolumn1.
describe formatted altercolumn1;
                            Describe Formatted Table
-------------------------------------------------------------------------------------------
 # col_name      data_type      comment
 a      integer
 name      varchar

 # Detailed Table Information
 Database:                default
 Owner:                   admintest
 LastAccessTime:          0
 Location:                hdfs://hacluster/user/hive/warehouse/altercolumn1
 Table Type:              MANAGED_TABLE
```



```
# Table Parameters:
        STATS_GENERATED_VIA_STATS_TASK  workaround for potential lack of HIVE-12730
        numFiles                0
        numRows                 0
        orc.compress.size       262144
        orc.compression.codec   GZIP
        orc.row.index.stride    10000
        orc.stripe.size         67108864
        presto_query_id         20210325_025238_00034_f63xj@default@HetuEngine
        presto_version
        rawDataSize             0
        totalSize               0
        transient_lastDdlTime   1616640758

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            25
Bucket Columns:         [a, name]
Sort Columns:           [SortingColumn{columnName=name, order=ASCENDING}]
Storage Desc Params:
        serialization.format    1
(1 row)

Query 20210325_090522_00091_f63xj@default@HetuEngine, FINISHED, 1 node
Splits: 1 total, 1 done (100.00%)
0:00 [0 rows, 0B] [0 rows/s, 0B/s]
```

11.13.2.12 ALTER VIEW

Syntax

- ALTER VIEW view_name AS select_statement;
- ALTER VIEW view_name SET TBLPROPERTIES table_properties;

Description

ALTER VIEW view_name AS select_statement; is used to change the definition of an existing view. Its syntax effect is similar to that of **CREATE OR REPLACE VIEW**.

The format of **table_properties** in **ALTER VIEW view_name SET TBLPROPERTIES table_properties;** is *(property_name = property_value, property_name = property_value, ...)*.

A view can contain the **Limit** and **ORDER BY** clauses. If the query statement of the associated view also contains these clauses, the final execution result is obtained from the calculation based on the clauses of the view. For example, if view V is specified to return five pieces of data and the associated query result is **select * from V limit 10**, only five pieces of data are returned.

Remarks

The preceding two syntax statements cannot be used together.

When a view contains partitions, the definition cannot be changed by using this syntax.

Example

```
CREATE OR REPLACE VIEW tv_view as SELECT id,name from (values (1, 'HetuEngine')) as x(id,name);

SELECT * FROM tv_view;
 id | name
----|------
  1 | HetuEngine
(1 row)

ALTER VIEW tv_view as SELECT id, brand FROM (VALUES (1, 'brand_1', 100), (2, 'brand_2', 300) ) AS x (id,
brand, price);

SELECT * FROM tv_view;
 id |  brand
----|---------
  1 | brand_1
  2 | brand_2
(2 rows)

ALTER VIEW tv_view SET TBLPROPERTIES ('comment' = 'This is a new comment');

show tblproperties tv_view;
                          SHOW TBLPROPERTIES
------------------------------------------------------------------------------
        comment                 'This is a new comment'
        presto_query_id         '20210325_034712_00040_f63xj@default@HetuEngine'
        presto_version
        presto_view             'true'
        transient_lastDdlTime
'1616644032'
(1 row)
```

11.13.2.13 ALTER SCHEMA

Syntax

ALTER (DATABASE|SCHEMA) schema_name SET LOCATION hdfs_location

ALTER (DATABASE|SCHEMA) database_name SET OWNER USER username

ALTER (DATABASE|SCHEMA) database_name SET DBPROPERTIES (property_name=property_value, ...);

Description

This statement does not move the current content of SCHEMA to the new path or modify the table or partition associated with the specified schema. It only modifies the upper-level directory of the table that is newly added to the database.

Example

```
Create schema foo;
-- Change the schema storage path.
ALTER SCHEMA foo SET LOCATION 'hdfs://hacluster/newlocation';
-- Change the owner of the schema.
ALTER SCHEMA foo SET OWNER user admin;
```

11.13.2.14 DROP SCHEMA

Syntax

DROP (DATABASE|SCHEMA) [IF EXISTS] databasename [RESTRICT|CASCADE]

Description

DATABASE and **SCHEMA** are conceptually equivalent and interchangeable.

This syntax is used to delete the database name. If the target database does not exist, an error message is displayed. However, if the **IF EXISTS** clause is used, no error message is displayed.

The optional parameter **RESTRICT|CASCADE** is used to specify the deletion mode. The default mode is **RESTRICT**. In this mode, the database can be deleted only when it is empty without any table. In **CASCADE** mode, tables in the database are deleted first, and then the database.

Example

- To drop the schema **web**.
  ```
  DROP SCHEMA web;
  ```
- If the schema **sales** exists, drop the schema:
  ```
  DROP SCHEMA IF EXISTS sales;
  ```
- Delete **schema test_drop** in cascading mode. The **tb_web** table that exists in **schema test_drop** is deleted before **schema test_drop** is deleted.
  ```
  CREATE SCHEMA test_drop;

  USE test_drop;

  CREATE TABLE tb_web(col1 int);

  DROP DATABASE test_drop CASCADE;
  ```

11.13.2.15 DROP TABLE

Syntax

DROP TABLE [IF EXISTS] table_name

Description

This statement is used to drop an existing table.

If the optional parameter IF EXISTS is specified and the table to be deleted does not exist, no error is reported.

The deleted data rows are moved to the recycle bin of HDFS.

Example

```
create table testfordrop(name varchar);
drop table if exists testfordrop;
```

11.13.2.16 DROP VIEW

Syntax

DROP VIEW [IF EXISTS] view_name

Description

This statement is used to delete an existing view. If the optional parameter IF EXISTS is specified and the view to be deleted does not exist, no error is reported.

Example

- Creating a View
  ```
  create view orders_by_date as select * from orders;
  ```
- Delete the **orders_by_date** view. If the view does not exist, an error is reported.
  ```
  DROP VIEW orders_by_date;
  ```
- Delete the **orders_by_date** view. Use the **IF EXISTS** parameter. If the view exists, it will be deleted. If the view does not exist, no error will be reported.
  ```
  DROP VIEW IF EXISTS orders_by_date;
  ```

11.13.2.17 DROP FUNCTION

Syntax

DROP FUNCTION [IF EXISTS] qualified_function_name

Description

It is used to delete an existing function that matches the given function name. If no matching function exists, the optional **IF EXISTS** clause will cause the **NOT_FOUND** error to be suppressed.

Example

- Delete the **example.namespace01.date_diff** function:

  DROP FUNCTION example.namespace01.date_diff
- Delete the **example.namespace01.date_diff** function if it exists:

  DROP FUNCTION IF EXISTS example.namespace01.date_diff

11.13.2.18 DROP MATERIALIZED VIEW

Syntax

DROP MATERIALIZED VIEW [IF EXISTS] view_name

Description

This statement is used to delete a materialized view. If the view to be deleted does not exist and the **if exists** parameter is used, no error occurs.

If a materialized view is deleted, metadata and table data associated with it will also be deleted.

NOTE

A materialized view will fail to delete if some data (metadata or table data) has been deleted before the materialized view is deleted.

Example

- Create a table.
```
hetuengine:tpcds_2gb> create table t1 (id int, c1 varchar);
hetuengine:tpcds_2gb> Insert into t1 values (1,'abc'), (2,'abc2'), (3,'abc3'), (4,'abc4'), (5,'abc5'), (6,'abc6');
```
- Create a materialized view.
```
hetuengine:tpcds_2gb> create materialized view mv.tpcds.t1 as select c1 from t1 where id <7;
```
- Delete a specified view. If the view does not exist, an error occurs.
```
hetuengine:tpcds_2gb> drop materialized view mv.tpcds.t1;
Query 20211206_095415_00003_k4wwu failed: line 1:1: MATERIALIZED VIEW 'mv.tpcds.t1' does not exist
```
- Delete a specified materialized view with the **if exists** parameter set. If the view exists, it will be deleted. If the view does not exist, no error occurs.
```
hetuengine:tpcds_2gb> drop materialized view if exists mv.tpcds.t1;
DROP MATERIALIZED VIEW
```

11.13.2.19 REFRESH MATERIALIZED VIEW

Syntax

REFRESH MATERIALIZED VIEW materialized_view_name

Description

This statement is used to update data in a materialized view.

You need to obtain the materialized view name in advance.

The following is an example.

Example

```
hetuengine:tpcds_orc_hive_2> refresh materialized view mv.tpcds.test;
REFRESH MATERIALIZED VIEW: 10 rows
```

11.13.2.20 TRUNCATE TABLE

Syntax

TRUNCATE [TABLE] table_name [PARTITION partition_spec];

partition_spec:

: (partition_column = partition_col_value, partition_column = partition_col_value, ...)

Description

This statement is used to remove all rows from a table or partition. You can use **partition_spec** to delete multiple specified partitions of a partitioned table at a time. If this parameter is not specified, all partitions of the partitioned table are deleted at a time. When table property **auto.purge** is set to **false** by default, deleted data rows are stored in the recycle bin of the file system. Otherwise, the data rows are directly deleted.

Remarks

The target table must be a control table, which means table propery **external** is set to **false**. Otherwise, an error will be reported during statement execution.

Example

```
-- Delete a native or control table
Create table simple(id int, name string);

Insert into simple values(1,'abc'),(2,'def');

select * from simple;
 id | name
----|------
  1 | abc
  2 | def
(2 rows)

Truncate table simple;

select * from simple;
 id | name
----|------
(0 rows)

-- Delete a table partition.
Create table tb_truncate_part (id int, name string) partitioned by (age int, state string);

Insert into tb_truncate_part values (1,'abc',10,'ap'),(2,'abc',10,'up'),(3,'abc',20,'ap'),(4,'abc',20,'up');

select * from tb_truncate_part;
 id | name | age | state
----|------|-----|-------
  2 | abc  |  10 | up
  3 | abc  |  20 | ap
  1 | abc  |  10 | ap
  4 | abc  |  20 | up
(4 rows

Truncate table tb_truncate_part partition (state = 'ap', age = 10);

select * from tb_truncate_part;
id | name | age | state
----|------|-----|-------
  4 | abc  |  20 | up
  2 | abc  |  10 | up
  3 | abc  |  20 | ap
(3 rows)
```

11.13.2.21 COMMENT

Syntax

COMMENT ON TABLE name IS 'comments'

Description

This statement is used to set the comment information of a table. You can delete the comment by setting the comment information to **NULL**.

Example

Change the comment of the **users** table to "master table". You can run the **show create table tablename** statement to view the comment.

```
COMMENT ON TABLE users IS 'master table';
```

11.13.2.22 VALUES

Syntax

VALUES row [, ...]

where row is a single expression or

(column_expression [, ...])

Description

VALUES is used to query any place that can be used (for example, the FROM clause of SELECT and INSERT). VALUES is used to create an anonymous table without column names, but tables and columns can be named using AS clauses with column aliases.

Example

- To return a table with one column and three rows:
  ```
  VALUES 1, 2, 3
  ```
- To return a table with two columns and three rows:
  ```
  VALUES
  (1, 'a'),
  (2, 'b'),
  (3, 'c')
  ```
- To return the table with the column name **id** and **name**:
  ```
  SELECT * FROM (values (1, 'a'), (2, 'b'),(3, 'c')) AS t (id, name);
  ```
- Create a table with the column **id** and **name**:
  ```
  CREATE TABLE example AS
   SELECT * FROM (VALUES (1, 'a'), (2, 'b'), (3, 'c')) AS t (id, name);
  ```

11.13.2.23 SHOW Syntax Overview

The **SHOW** syntax is used to view information about database objects. The **LIKE** clause is used to filter database objects. The following lists the matching rules. For details, see **SHOW TABLES**.

Rule 1: Underscores (_) can be used to match any single character.

Rule 2: Percent signs (%) can be used to match zero or any number of characters.

Rule 3: Asterisks (*) can be used to match zero or any number of characters.

Rule 4: Vertical bars (|) can be used to separate multiple rules.

Rule 5: To use an underscore (_) as a matching condition, use **ESCAPE** to specify an escape character to escape the underscore (_) so that it will not be parsed according to rule 1.

11.13.2.24 SHOW CATALOGS

Syntax

SHOW CATALOGS [LIKE pattern [ESCAPE escapeChar]]

Description

This expression is used to list available catalogs. The optional parameter **like** is used for keyword-based matching.

Example

- List all catalogs:
  ```
  SHOW CATALOGS;
  ```
- List all catalogs whose name prefix is **sys**:
  ```
  SHOW CATALOGS LIKE 'sys%';
  ```

11.13.2.25 SHOW SCHEMAS (DATABASES)

Syntax

SHOW SCHEMAS|DATABASES [(FROM| IN) catalog] [LIKE pattern [ESCAPE escapeChar]]

Description

In this statement, **DATABASES** and **SCHEMAS** are conceptually equivalent and interchangeable. This statement is used to list all schemas defined in MetaStore. The optional clause **LIKE** can use rule operations to filter results. It supports the wildcards '*' (matching any character) and '|' (matching optional items).

Example

List all schemas of the current catalog:

```
SHOW SCHEMAS;
```

List all schemas whose schema name prefix is **t** in a specified catalog:

```
SHOW SCHEMAS FROM hive LIKE 't%';

--Equivalent writing:
SHOW SCHEMAS IN hive LIKE 't%';
```

If a character in the matching character string conflicts with the wildcard, you can specify an escape character to identify the character. For example, to query all schemas whose schema name prefix is **pm_** in the **hive** catalog, set the escape character to **/**.

```
SHOW SCHEMAS IN hive LIKE 'pm/_%' ESCAPE '/';
```

11.13.2.26 SHOW TABLES

Syntax

SHOW TABLES [(FROM | IN) schema] [LIKE pattern [ESCAPE escapeChar]]

Description

This expression is used to list all tables in a specified schema. If no schema is specified, the current schema is used by default.

The optional parameter **like** is used for keyword-based matching.

Example

```
--Create a test table.
Create table show_table1(a int);
Create table show_table2(a int);
Create table showtable5(a int);
Create table intable(a int);
Create table fromtable(a int);

-- Match a single character '_'.
show tables in  default like 'show_table_';
  Table
-------------
 show_table1
 show_table2
(2 rows)

-- Match multiple characters '*' and '%'.
show tables in default like 'show%';
Table
-------------
 show_table1
 show_table2
 showtable5
(3 rows)

show tables in default like 'show*';
Table
-------------
 show_table1
 show_table2
 showtable5
(3 rows)

-- The escape character is used. In the second example, '_' is used as the filter condition. The result set does not contain showtable5.
show tables in default like 'show_%';
    Table
-------------
 show_table1
 show_table2
 showtable5
(3 rows)

show tables in default like 'show$_%' ESCAPE '$';
    Table
-------------
 show_table1
 show_table2
(2 rows)

-- If multiple conditions are met, query the tables whose names start with show_ or in in default.
show tables in default like 'show$_%|in%' ESCAPE '$';
    Table
-------------
 intable
 show_table1
 show_table2
(3 rows)
```


11.13.2.27 SHOW TBLPROPERTIES TABLE|VIEW

Syntax

SHOW TBLPROPERTIES table_name|view_name[(property_name)]

Description

View the properties of a table or the properties of a keyword.

If you do not specify an attribute keyword, this statement returns all table attributes.

Otherwise, this statement returns the attribute that matches the specified keyword.

Example

```
-- View all table attributes of show_table1.
                        SHOW TBLPROPERTIES
-------------------------------------------------------------------------------
       STATS_GENERATED_VIA_STATS_TASK  'workaround for potential lack of HIVE-12730'
       auto.purge              'false'
       numFiles                '0'
       numRows                 '0'
       orc.compress.size       '262144'
       orc.compression.codec   'GZIP'
       orc.row.index.stride    '10000'
       orc.stripe.size         '67108864'
       presto_query_id         '20230909_095107_00042_2hwbg@default@HetuEngine'
       presto_version          '399'
       rawDataSize             '0'
       totalSize               '0'
       transient_lastDdlTime   '1694253067'

(1 row)

-- View the compression algorithm of show_table1.
SHOW TBLPROPERTIES show_table1('orc.compression.codec');
SHOW TBLPROPERTIES
---------------------
 GZIP
(1 row)
```

11.13.2.28 SHOW TABLE/PARTITION EXTENDED

Syntax

SHOW TABLE EXTENDED [IN | FROM schema_name] LIKE 'identifier_with_wildcards' [PARTITION (partition_spec)]

Description

This statement is used to display details about a table or partition.

Regular expressions can be used to match multiple tables at the same time, but cannot be used to match partitions.

The displayed information includes basic table information and file system information. The file system information includes the total number of files, total

file size, maximum file length, minimum file length, last access time, and last update time. For a specified partition, its file system information is provided, instead of the file system information of the table where the partition is located.

Parameters

- IN | FROM schema_name

  This parameter specifies the schema name. If this parameter is not specified, the current schema is used by default.

- LIKE 'identifier_with_wildcards'

  identifier_with_wildcards supports only rule matching expressions that contain asterisks (*) and vertical bars (|).

  The asterisk (*) can match one or more characters. The vertical bar (|) separates multiple matching rules.

  Spaces at the beginning and end of a rule matching expression are not used for matching.

- partition_spec

  This parameter is optional. It specifies the partition list using key pairs. Multiple key pairs are separated by commas. Table names do not support fuzzy match when partitions are specified.

Example

```
-- Data preparation
create schema show_schema;

use show_schema;

create table show_table1(a int,b string);
create table show_table2(a int,b string);
create table from_table1(a int,b string);
create table in_table1(a int,b string);

-- Query the detailed information about tables whose names start with "show".
show table extended  like 'show*';
                            tab_name
--------------------------------------------------------------------------------
 tableName:show_table1
 owner:admintest
 location:hdfs://hacluster/user/hive/warehouse/show_schema.db/show_table1
 InputFormat:org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
 OutputFormat:org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
 columns:struct columns {int a,string b}
 partitioned:false
 partitionColumns:
 totalNumberFiles:0
 totalFileSize:0

 tableName:show_table2
 owner:admintest
 location:hdfs://hacluster/user/hive/warehouse/show_schema.db/show_table2
 InputFormat:org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
 OutputFormat:org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
 columns:struct columns {int a,string b}
 partitioned:false
 partitionColumns:
 totalNumberFiles:0
 totalFileSize:0

(1 row)
```



```
-- Query the detailed information about tables whose names start with "from" or "show".
 show table extended  like 'from*|show*';
                            tab_name
--------------------------------------------------------------------------
 tableName           show_table1
 owner               admintest
 location            hdfs://hacluster/user/hive/warehouse/show_table1
 InputFormat           org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
 OutputFormat          org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
 columns             struct columns {int a,string b}
 partitioned         false
 partitionColumns
 totalNumberFiles    0
 totalFileSize       null

 tableName           from_table1
 owner               admintest
 location            hdfs://hacluster/user/hive/warehouse/from_table1
 InputFormat           org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
 OutputFormat          org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
 columns             struct columns {int a,string b}
 partitioned         false
 partitionColumns
 totalNumberFiles    0
 totalFileSize       null

 tableName           show_table2
 owner               admintest
 location            hdfs://hacluster/user/hive/warehouse/show_table2
 InputFormat           org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
 OutputFormat          org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
 columns             struct columns {int a,string b}
 partitioned         false
 partitionColumns
 totalNumberFiles    0
 totalFileSize       null

(1 row)
-- Query the detailed information about the page_views table in the web schema.
 show table extended from web like 'page*';
                               tab_name
-------------------------------------------------------------------------------
 tableName:page_views
 owner:admintest
 location:hdfs://hacluster/user/web.db/page_views
 InputFormat:org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
 OutputFormat:org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
 columns:struct columns {timestamp view_time,bigint user_id,string page_url}
 partitioned:true
 partitionColumns: struct partition_columns {date ds,string country}
 totalNumberFiles:0
 totalFileSize:0

(1 row)
```

11.13.2.29 SHOW STATS

Syntax

SHOW STATS FOR table_name;

SHOW STATS FOR (SELECT * FROM table [WHERE condition]);

Remarks

ANALYZE should be performed on the table before SHOW STATS. For details, see **ANALYZE**. If you run the **show stats** command on the target table before running ANALYZE, all values are null.

Description

Returns the approximate statistics of a table.

Returns the statistics about each column.

Column	Description
column_name	Column name (Summary Row: **NULL**)
data_size	The total size of all values in the column (in bytes)
distinct_values_count	Number of different values in the column
nulls_fraction	Fractions whose values are **NULL** in the column
row_count	Number of lines (returned only for summary lines)
low_value	Minimum value found in this column (only for some types)
high_value	Maximum value found in this column (applicable only to some types)

Example

```
SHOW STATS FOR orders;
SHOW STATS FOR (SELECT * FROM orders);
```

- Before analyzing the **nation** table:

```
SHOW STATS FOR nation;
column_name | data_size | distinct_values_count | nulls_fraction | row_count | low_value | high_value
-------------|-----------|-----------------------|----------------|-----------|-----------|------------
 name        |      NULL |                  NULL |           NULL |      NULL | NULL      | NULL
 regionkey   |      NULL |                  NULL |           NULL |      NULL | NULL      | NULL
 NULL        |      NULL |                  NULL |           NULL |       6.0 | NULL      | NULL
(3 rows)
```

- After analyzing the **nation** table:

```
Analyze nation;
ANALYZE: 6 rows

-- Query the analysis result.
SHOW STATS FOR nation;
column_name | data_size | distinct_values_count | nulls_fraction | row_count | low_value | high_value
-------------|-----------|-----------------------|----------------|-----------|-----------|------------
 name        |      45.0 |                   5.0 |            0.0 |      NULL | NULL      | NULL
 regionkey   |      NULL |                   2.0 |            0.0 |      NULL | 0         | 2
 NULL        |      NULL |                  NULL |           NULL |       6.0 | NULL      | NULL
(3 rows)
```


11.13.2.30 SHOW FUNCTIONS

Syntax

SHOW FUNCTIONS [LIKE pattern [ESCAPE escapeChar]];

SHOW EXTERNAL FUNCTIONS;

SHOW EXTERNAL FUNCTION qualified_function_name;

Description

Displays the definitions of all built-in functions.

Displays the description of all Java functions.

Displays the definition of a specified function.

Example

```
SHOW functions;

-- Use the LIKE clause.
show functions like 'boo_%';
 Function | Return Type | Argument Types | Function Type | Deterministic | Description
----------|-------------|----------------|---------------|---------------|-------------
 bool_and | boolean     | boolean        | aggregate     | true          |
 bool_or  | boolean     | boolean        | aggregate     | true          |
(2 rows)

-- If a character in the matching character string conflicts with the wildcard, you can specify an escape character to identify the character. For example, to query all tables whose table name prefix is t_ in the default schema, set the escape character to \.
SHOW FUNCTIONS LIKE 'array\_%' escape '\';
  Function     | Return Type |        Argument Types          | Function Type | Deterministic
|                                        Description                                         | Variable Arity | Built In
----------------|-------------|--------------------------------|---------------|---------------
|-----------------------------------------------------------------------------------------------------------------------|----------------|----
------
 array_agg       | array(T)    | T                              | aggregate     | true
| return an array of values                                                                                             | false          | true
 array_contains  | boolean     | array(T), T                    | scalar        | true
| Determines whether given value exists in the array                                                                    | false          |
true
 array_distinct  | array(E)    | array(E)                       | scalar        | true
| Remove duplicate values from the given array                                                                          | false          |
true
 array_except    | array(E)    | array(E), array(E)             | scalar        | true
| Returns an array of elements that are in the first array but not the second, without duplicates.                      |
false          | true
 array_intersect | array(E)    | array(E), array(E)             | scalar        | true
| Intersects elements of the two given arrays                                                                           | false          | true
 array_join      | varchar     | array(T), varchar              | scalar        | true
| Concatenates the elements of the given array using a delimiter and an optional string to replace nulls |
false          | true
 array_join      | varchar     | array(T), varchar, varchar     | scalar        | true
| Concatenates the elements of the given array using a delimiter and an optional string to replace nulls |
false          | true
 array_max       | T           | array(T)                       | scalar        | true
| Get maximum value of array                                                                                            | false          | true
 array_min       | T           | array(T)                       | scalar        | true
| Get minimum value of array                                                                                            | false          | true
 array_position  | bigint      | array(T), T                    | scalar        | true
| Returns the position of the first occurrence of the given value in array (or 0 if not found)                          |
false          | true
 array_remove    | array(E)    | array(E), E                    | scalar        | true
```



```
| Remove specified values from the given array                                    | false       | true
 array_sort     | array(E)    | array(E)                    | scalar       | true
| Sorts the given array in ascending order according to the natural ordering of its elements.          |
false          | true
 array_sort     | array(T)    | array(T), function(T,T,integer) | scalar       | true
| Sorts the given array with a lambda comparator.                                  | false          |
true
 array_union    | array(E)    | array(E), array(E)            | scalar       | true
| Union elements of the two given arrays                                           | false          | true

-- View all Java functions.
SHOW external functions;
  Function                   |   Owner
---------------------------|-----------
 example.namespace02.repeat | admintest
 hetu.default.add_two       | admintest
(2 rows)

-- View the definition of a specified function.
SHOW external function example.namespace02.repeat;
         External Function
-------------------------------------
External FUNCTION example.namespace02.repeat (
    s varchar,
    n integer
 )
 RETURNS varchar

 COMMENT 'repeat'
 LANGUAGE JAVA
 DETERMINISTIC
 CALLED ON NULL INPUT
 SYMBOL com.test.udf.hetuengine.functions.repeat
 URI hdfs://hacluster/user/hetuserver/udf/data/hetu_udf/udf-test-0.0.1-SNAPSHOT.jar

 FUNCPROPERTIES (
    owner = 'admintest'
 )
```

11.13.2.31 SHOW SESSION

- Syntax

  SHOW SESSION;

- Description

  This expression is used to list the configuration parameters of all sessions.

- Example

```
show session;
```

11.13.2.32 SHOW PARTITIONS

- Syntax

  SHOW PARTITIONS [catalog_name.][db_name.]table_name [PARTITION (partitionSpecs)];

- Description

  This statement is used to list all specified partitions.

- Example

```
SHOW PARTITIONS test PARTITION(hr = '12', ds = 12);
SHOW PARTITIONS test PARTITION(ds > 12);
```


11.13.2.33 SHOW COLUMNS

Syntax

SHOW COLUMNS [FROM | IN] table

Description

This expression is used to list the column information of a specified table.

Example

List the column information of the **fruit** table:

```
SHOW COLUMNS FROM fruit;
SHOW COLUMNS IN fruit;
```

11.13.2.34 SHOW CREATE TABLE

Syntax

SHOW CREATE TABLE table_name

Description

This statement is used to display the SQL creating statement for a specified data table.

Example

To display the SQL statements that can be used to create the table **orders**:

```
CREATE TABLE orders (
 orderkey bigint,
 orderstatus varchar,
 totalprice double,
 orderdate date
 )
 WITH (format = 'ORC', location='/user',orc_compress='ZLIB',external=true, "auto.purge"=false);

show create table orders;

                  Create Table
-----------------------------------------------------
 CREATE TABLE hive.default.orders (
   orderkey bigint,
   orderstatus varchar,
   totalprice double,
   orderdate date
 )
 WITH (
   external_location = 'hdfs://hacluster/user',
   format = 'ORC',
   orc_compress = 'ZLIB',
   orc_compress_size = 262144,
   orc_row_index_stride = 10000,
   orc_stripe_size = 67108864
 )
(1 row)
```


11.13.2.35 SHOW VIEWS

Syntax

SHOW VIEWS [IN/FROM database_name] [LIKE pattern [ESCAPE escapeChar]]

Description

This statement is used to list all views that meet the conditions in a specified schema.

By default, the current schema is used. You can also use the **in/from** clause to specify a schema.

The **LIKE** clause is used to filter views whose names meet the regular expression. If this clause is not used, all views are listed. The matched views are sorted in alphabetic order.

Currently, the regular expression supports only the asterisk (*) (matching any character).

Example

Create views:

```
Create schema test1;
Use test1;
Create table t1(id int, name string);
Create view v1 as select * from t1;
Create view v2 as select * from t1;
Create view t1view as select * from t1;
Create view t2view as select * from t1;

Show views;
Table
--------
 t1view
 t2view
 v1
 v2
(4 rows)

Show views like 'v1';
 Table
-------
 v1
(1 row)

Show views 'v_';
Table
-------
 v1
 v2
(2 rows)
show views like 't*';
 Table
--------
 t1view
 t2view

Show views in test1;
 Table
--------
 t1view
```



```
 t2view
 v1
 v2
(4 rows)
```

11.13.2.36 SHOW CREATE VIEW

Syntax

SHOW CREATE VIEW view_name

Description

This statement is used to display the SQL creation statement of a specified data view.

Example

To display the SQL statement that can be used to create the **order_view** view:

```
SHOW CREATE VIEW test_view;
             Create View
----------------------------------------
 CREATE VIEW hive.default.test_view AS
 SELECT
   orderkey
 , orderstatus
 , (totalprice / 4) quarter
 FROM
   orders
(1 row)
```

11.13.2.37 SHOW MATERIALIZED VIEWS

Syntax

SHOW MATERIALIZED VIEWS [IN/FROM schema_name] [WITH TABLES LIKE pattern]

Description

This statement is used to list all materialized views whose **catalogName** is **mv** and corresponding data tables. If you want to view only the MVs in a schema, use the **[IN/FROM schema_name]** clause.

The **LIKE** clause is used to filter views whose names meet the regular expression. If this clause is not used, all views are listed. The matched views are sorted in alphabetic order.

Currently, the regular expression supports the following matching rules: asterisk (*) or percent sign (%) for matching any character, underscore (_) for matching one character, and vertical bar (|) for connecting two or more conditions.

Example

- SHOW MATERIALIZED VIEWS;

  ```
  hetuengine:tpcds_2gb> SHOW MATERIALIZED VIEWS;
                Materialized Views                |     Last Refresh Time      | Status | State | SyncStatus
  ```

```
-----------------------------------------------------------|-------------------------------|--------|-------|------------
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_1   | Wed Sep 07 15:37:39 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_103 | Wed Sep 07 15:31:41 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_15  | Wed Sep 07 14:47:09 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_16  | Wed Sep 07 15:39:09 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_18  | Wed Sep 07 14:46:42 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_23  | Wed Sep 07 15:39:49 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_24  | Wed Sep 07 14:48:19 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_28  | Wed Sep 07 15:38:38 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_29  | Wed Sep 07 15:38:19 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_30  | Wed Sep 07 15:38:04 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_31  | Wed Sep 07 15:39:28 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_33  | Wed Sep 07 15:39:39 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_34  | Wed Sep 07 14:47:38 CST 2022 |
ENABLE | Stale | true
```

NOTE

- **Materialized Views**: name of a materialized view
- **Last Refresh Time**: last time when the materialized view is refreshed
- **Status**: materialized view status
  - **DISABLE**: The materialized view is unavailable because it fails to be automatically refreshed for three consecutive times. The materialized view cannot be rewritten.
  - **ENABLE**: normal state
  - **REFRESHING**: refreshing
  - **INIT**: The materialized view is created for the first time and has no entity data.
  - **SUSPEND**: suspended state, which cannot be rewritten or refreshed
- **State**: validity period of the materialized view
  - **Stale**: The materialized view expires.
  - **Valid**: The materialized view does not expire and is in the normal state.
- **SyncStatus**: whether the materialized view cache is synchronized.

- SHOW MATERIALIZED VIEWS FROM tpcds;

```
hetuengine:tpcds_2gb> SHOW MATERIALIZED VIEWS FROM auto_created_mv;
                Materialized Views                |      Last Refresh Time       | Status | State | SyncStatus
-----------------------------------------------------------|-------------------------------|--------|-------|------------
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_1   | Wed Sep 07 15:37:39 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_103 | Wed Sep 07 15:31:41 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_15  | Wed Sep 07 14:47:09 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_16  | Wed Sep 07 15:39:09 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_18  | Wed Sep 07 14:46:42 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_23  | Wed Sep 07 15:39:49 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_24  | Wed Sep 07 14:48:19 CST 2022 |
ENABLE | Stale | true
```



```
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_28  | Wed Sep 07 15:38:38 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_29  | Wed Sep 07 15:38:19 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_30  | Wed Sep 07 15:38:04 CST 2022 |
ENABLE | Stale | true
```

- SHOW MATERIALIZED VIEWS WITH TABLES LIKE 'hive.tpcds_bin_partitioned_orc_2.call_center';

```
hetuengine:tpcds_2gb> SHOW MATERIALIZED VIEWS WITH TABLES LIKE
'hive.tpcds_bin_partitioned_orc_2.call_center';
                Materialized Views                |                Tables                |     Last Refresh
Time      | Status | State | SyncStatus
------------------------------------------------------|----------------------------------------------|--------------
----------------|--------|-------|------------
 mv.auto_created_mv.auto_created_mv_20220907_142132_mv_65 |
hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:28:20 CST 2022 | ENABLE | Stale |
true
 mv.auto_created_mv.auto_created_mv_20220907_152143_mv_19 |
hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:37:07 CST 2022 | ENABLE | Stale | true
```

- SHOW MATERIALIZED VIEWS WITH TABLES LIKE '_ive.tpcds_bin_partitioned_orc_2.call_center';

```
hetuengine:tpcds_2gb> SHOW MATERIALIZED VIEWS WITH TABLES LIKE
'_ive.tpcds_bin_partitioned_orc_2.call_center';
                Materialized Views                |                Tables                |     Last Refresh
Time      | Status | State | SyncStatus
------------------------------------------------------|----------------------------------------------|--------------
----------------|--------|-------|------------
 mv.auto_created_mv.auto_created_mv_20220907_142132_mv_65 |
hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:28:20 CST 2022 | ENABLE | Stale |
true
 mv.auto_created_mv.auto_created_mv_20220907_152143_mv_19 |
hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:37:07 CST 2022 | ENABLE | Stale | true
```

- SHOW MATERIALIZED VIEWS TABLES LIKE '*.call_center';

```
hetuengine:tpcds_2gb> SHOW MATERIALIZED VIEWS WITH TABLES LIKE '*.call_center';
                Materialized Views                |                Tables                |     Last Refresh
Time      | Status | State | SyncStatus
------------------------------------------------------|----------------------------------------------|--------------
----------------|--------|-------|------------
 mv.auto_created_mv.auto_created_mv_20220907_142132_mv_65 |
hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:28:20 CST 2022 | ENABLE | Stale |
true
 mv.auto_created_mv.auto_created_mv_20220907_152143_mv_19 |
hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:37:07 CST 2022 | ENABLE | Stale | true
```

- SHOW MATERIALIZED VIEWS WITH TABLES LIKE '*.**call_center**|*.date_dim';

```
hetuengine:tpcds_2gb> SHOW MATERIALIZED VIEWS WITH TABLES LIKE '*.call_center|*.date_dim';
                Materialized Views                |                Tables                |     Last Refresh
Time      | Status | State | SyncStatus
------------------------------------------------------|----------------------------------------------|--------------
----------------|--------|-------|------------
 mv.auto_created_mv.auto_created_mv_20220907_142132_mv_65 |
hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:28:20 CST 2022 | ENABLE | Stale |
true
 mv.auto_created_mv.auto_created_mv_20220907_152143_mv_19 |
hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:37:07 CST 2022 | ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_1  |
hive.tpcds_bin_partitioned_orc_2.date_dim    | Wed Sep 07 15:37:39 CST 2022 | ENABLE | Stale |
true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_103|
hive.tpcds_bin_partitioned_orc_2.date_dim    | Wed Sep 07 15:31:41 CST 2022 | ENABLE | Stale | true
```

11.13.2.38 SHOW CREATE MATERIALIZED VIEW

Syntax

SHOW CREATE MATERIALIZED VIEW materialized_view_name

Description

This statement is used to display the SQL statements that can be used to create a materialized view.

Example

Displays the SQL statements for creating the materialized view.

```
hetuengine:tpcds_2gb> show create materialized view mv.tpcds.test;
            Create Materialized View
-----------------------------------------------------
 CREATE MATERIALIZED VIEW mv.tpcds.test ( c1, id )
 WITH (
    storage_table = 'mppdb.tpcds.test'
 ) AS
 SELECT
   c1
 , id
 FROM
   t1
WHERE (id < 7)
```

11.13.3 HetuEngine DML SQL Syntax

11.13.3.1 INSERT

Syntax

INSERT { INTO | OVERWRITE } [TABLE] table_name [(column_list)] [PARTITION (partition_clause)] {select_statement | VALUES (value [, value ...]) [, (value [, value ...]) ...] }

FROM from_statement INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement

FROM from_statement INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...) select_statement

Remarks

If there is only one field in the data table and the field type can be **row** or **struct**, use **row** to encapsulate the type when inserting data.

```
-- When a complex type is inserted into a single field table, the complex type must be wrapped by row().
CREATE TABLE test_row (id row(c1 int, c2 string));

INSERT INTO test_row values row(row(1, 'test'));


--The complex type of a multi-field table can be directly inserted.
CREATE TABLE test_multy_value(id int, col row(c1 int, c2 string));

INSERT INTO test_multy_value values (1,row(1,'test'));
```

Description

- This statement is used to insert a new data row into a table.

- If a list of column names is specified, the list of column names must exactly match the name of the column list generated by the **query** statement. The value of each column that is not in the column name list is set to **null**.
- If no column name list is specified, the column generated by the query statement must exactly match the column to be inserted.
- When **INSERT INTO** is used, data is added to the table. When **INSERT OVERWRITE** is used, if table property **auto.purge** is set to **true**, data in the original table data is directly deleted and new data is written to the table.
- If the object table is a partitioned table, **insert overwrite** deletes the data in the corresponding partition instead of all data.
- The **table** keyword following INSERT INTO is optional to be compatible with the Hive syntax.

Example

- Create the **fruit** and **fruit_copy** tables.
```
create table fruit (name varchar,price double);
create table fruit_copy (name varchar,price double);
```
- Inserts a row of data into the **fruit** table.
```
insert into fruit values('LIchee',32);
--The following is an example of the compatible format with the table keyword:
insert into table fruit values('Cherry',88);
```
- Insert multiple lines of data into the **fruit** table.
```
insert into fruit values('banana',10),('peach',6),('lemon',12),('apple',7);
```
- Load the data lines in the **fruit** table to the **fruit_copy** table. After the execution, there are five records in the table.
```
insert into fruit_copy select * from fruit;
```
- Clear the **fruit_copy** table, and then load the data in the **fruit** table to the table. After the execution, there are two records in the **fruit_copy** table.
```
insert overwrite fruit_copy select *  from fruit limit 2;
```
- For the VARCHAR type, INSERT can be used only when the column length defined in the target table is greater than the actual column length of the source table. SELECT queries data from the source table and inserts the data to the target table.
```
create table varchar50(c1 varchar(50));
insert into varchar50 values('hetuEngine');
create table varchar100(c1 varchar(100));
insert into varchar100 select * from varchar50;
```
- When the **insert overwrite** statement is used for a partitioned table, only the data in the partition where the inserted value is located is cleared, not the entire table.
```
--Create a table.
create table test_part (id int, alias varchar) partitioned by (dept_id int, status varchar);

insert into test_part  partition(dept_id=10, status='good') values (1, 'xyz'), (2, 'abc');

select * from test_part order by id;
 id | alias | dept_id | status
----|-------|---------|--------
  1 | xyz   |      10 | good
  2 | abc   |      10 | good
(2 rows)

--Clear the partition(dept_id=25, status='overwrite') partition and insert a data record.
insert overwrite test_part (id, alias, dept_id, status) values (3, 'uvw', 25, 'overwrite');
 select * from test_part ;
 id | alias | dept_id |  status
```



```
----|-------|---------|-----------
 1 | xyz   |      10 | good
 2 | abc   |      10 | good
 3 | uvw   |      25 | overwrite

--Clear the partition(dept_id=10, status='good') partition and insert a data record.
insert overwrite test_part (id, alias, dept_id, status) values (4, 'new', 10, 'good');
select * from test_part ordr;
 id | alias | dept_id |  status
----|-------|---------|-----------
 3 | uvw   |      25 | overwrite
 4 | new   |      10 | good
(2 rows)

-- Insert data to a partitioned table.
create table test_p_1(name string, age int) partitioned by (provice string, city string);

 create table test_p_2(name string, age int) partitioned by (provice string, city string);

-- Add data to test_p_1.
 insert into test_p_1 partition (provice = 'hebei', city= 'baoding') values ('xiaobei',15),( 'xiaoming',22);
-- Insert data into test_p_2 based on test_p_1.

-- Method 1
from test_p_1 insert into table test_p_2 partition (provice = 'hebei', city= 'baoding') select name,age;

-- Method 2
insert into test_p_2 partition(provice = 'hebei', city= 'baoding') select name,age from test_p_1;
```

Precautions

By default, data cannot be inserted into external tables. To insert data into external tables, add configurations to the data source.

- Co-deployment

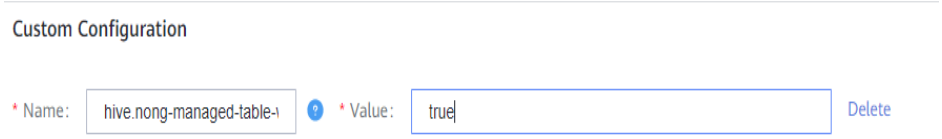
  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HetuEngine** > **Dashboard**, and click the HSConsole link on the **HSConsole Web UI** to go to the compute instance page.

  Choose **Data Source** > **Hive** > **Edit**. On the **Custom Configuration** page, click **Add** to add custom configuration item **hive.non-managed-table-writes-enabled** and set it to **true**.

- Independent deployment of Hive

  Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HetuEngine** > **Dashboard**, and click the HSConsole link on the **HSConsole Web UI** to go to the compute instance page.

  Choose **Data Source** > *Hive data source name* > **Edit**. On the **Custom Configuration** page, click **Add** to add custom configuration item **hive.non-managed-table-writes-enabled** and set it to **true**.



11.13.3.2 DELETE

Syntax

DELETE FROM table_name [WHERE condition]

Description

This statement is used to delete data from a table.

In the current version, you can run the **delete** command to delete the data of an entire table or a specified partition in a partition table.

For a transaction table (the attribute **transactional** is set to **true**), if the **where** condition is specified, the rows that match the condition are deleted.

Example

Non-transaction table scenario:

- Clear the table data.

```
--Create a table and insert data into the table.
 create table tb_del as select * from (values(1,'suse'),(2,'centos'),(3,'euler')) as t (id,os);
select * from tb_del;
 id |   os
----|--------
  1 | suse
  2 | centos
  3 | euler
(3 rows)

--A single data record cannot be deleted using the WHERE clause.
delete from tb_del where id =1;
Query 20201116_081955_00027_iyct5@default@HetuEngine failed: This connector only supports
delete where one or more partitions are deleted entirely for Non-Transactional tables

--Clear the table data.
delete from tb_del;
select * from tb_del;
 id | os
----|----
(0 rows)
```

- Delete the **partition(date='2020-07-17', country='US')** partition from the **hive.web.page_views** partition table.

```
delete from hive.web.page_views where ds=date '2020-07-17' and country='US';
```

Transaction table scenario: deleting a specified record

```
-- Create a transaction table.
create table tb_trans(a int,b string) with (transactional=true);
CREATE TABLE

-- Insert data.
insert into tb_trans values(1,'a'),(2,'b'),(3,'c');
INSERT: 3 rows

-- Delete data.
delete from tb_trans where a=1;
DELETE: 1 row
```


11.13.3.3 UPDATE

Syntax

UPDATE tablename SET column = value [, column = value ...] [WHERE expression]

Description

This statement is used to update table data based on conditions.

Remarks

- Only the transaction table in ORC format is supported, and the table cannot be an external table.
- The syntax **set (column_name1,column_name2, ...) = (value1,value2, ...)** is not supported.

Example

```
-- Create a transaction table.
create table upd_tb(col1 int,col2 string)  with (format='orc',transactional=true);

--Insert data.
insert into upd_tb values (3,'A'),(4,'B');

-- Change the value of col1 = 4.
update upd_tb set col1=5 where col1=4;

-- The col1=4 record is modified.
select * from upd_tb; --
 col1 | col2
------|------
    5 | B
    3 | A
```

11.13.3.4 LOAD

Syntax

LOAD DATA INPATH filepath [OVERWRITE] INTO TABLE tablename

[PARTITION(partcol1=value1,partcol2=values2...)]

Description

This statement is used to load data from a file or folder to a table.

Filepath: Enter the absolute path of the file or directory.

OVERWRITE: If this keyword is used, the data in the target table (or partition) is deleted and replaced with the data read from the file.

Remarks

To load data to a specified partition, you must list all fields of the table in the partition clause.

Complex data types, such as Array and Map, are not supported.

External tables are not supported.

The format of the data file must be the same as that of the target table.

When creating the target table, specify the separator of the file and ensure that the separator is the same as that in the data file.

Example

Create the **f1.txt** file, enter three lines of numbers, and upload the file to the **/opt/load_test/** directory by using HDFS.

```
-- Read data from the f1.txt file and fill in the f1 table.
CREATE TABLE tb_load_f1(id int) with (format='TEXTFILE');

LOAD DATA INPATH '/opt/load_test/f1.txt' into table tb_load_f1;

select * from tb_load_f1;
 id
----
  1
  2
  3
(3 rows)

-- Read files in the /opt/load_test/ directory and fill in the f2 table.
CREATE TABLE tb_load_f2(id int) with (format='TEXTFILE');

LOAD DATA INPATH '/opt/load_test/' into table tb_load_f2;

select * from tb_load_f2;
 id
----
  1
  2
  3
(3 rows)

-- Read the f3.txt file and fill in the f3 table (use hyphens (-) to separate multiple fields), and upload the f3.txt file to the /opt/load_test/ directory by using HDFS. The content of the f3.txt file is as follows:
 1-n1
 2-n2
-- Create the tb_load_f3 table.
CREATE TABLE tb_load_f3(id int,name varchar) with(format='TEXTFILE',textfile_field_separator='-');

Load data inpath '/opt/load_test/f3.txt' into table tb_load_f3;

Select * from tb_load_f3;
id | name
----|------
  1 | n1
  2 | n2
(2 rows)
```

11.13.4 HetuEngine TCL SQL Syntax

11.13.4.1 START TRANSACTION

Syntax

START TRANSACTION [mode [, ...]]

mode is used to set the transaction isolation level. The options are as follows:

ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE }

READ { ONLY | WRITE }

Description

This statement is used to start a new transaction in the current session.

Example

```
START TRANSACTION;
START TRANSACTION ISOLATION LEVEL REPEATABLE READ;
START TRANSACTION READ WRITE;
START TRANSACTION ISOLATION LEVEL READ COMMITTED, READ ONLY;
START TRANSACTION READ WRITE, ISOLATION LEVEL SERIALIZABLE;
```

NOTE

Nested transactions are not supported. That is, after a transaction is started, other transactions cannot be started before the transaction is committed.

11.13.4.2 COMMIT

- Syntax

  COMMIT [WORK]

- Description

  This statement is used to commit the current transaction.

- Example

  ```
  COMMIT;
  COMMIT WORK;
  ```

11.13.4.3 ROLLBACK

- Syntax

  ROLLBACK [WORK]

- Description

  This statement is used to roll back the current transaction.

- Example

  ```
  ROLLBACK;
  ROLLBACK WORK;
  ```

11.13.5 HetuEngine DQL SQL Syntax

11.13.5.1 SELECT

Syntax

[/*+ query_rewrite_hint*/]

[WITH [RECURSIVE] with_query [, ...]]

SELECT [ALL | DISTINCT] select_expression [, ...]

[FROM from_item [, ...]]

[WHERE condition]

[GROUP BY [ALL | DISTINCT] grouping_element [, ...]]

[HAVING condition]

[{ UNION | INTERSECT | EXCEPT } [ALL | DISTINCT] select]

[ORDER BY expression [ASC | DESC] [, ...]]

[OFFSET count [ROW | ROWS]]

[LIMIT { count | ALL }]

[FETCH { FIRST | NEXT } [count] { ROW | ROWS } { ONLY | WITH TIES }]

NOTE

- **from_item** can be used in the following formats:
  - table_name [[AS] alias [(column_alias [, ...])]]
  - from_item join_type from_item [ON join_condition | USING (join_column [, ...])]
  - table_name [[AS] alias [(column_alias [, ...])]]
    MATCH_RECOGNIZE pattern_recognition_specification
    [[AS] alias [(column_alias [, ...])]]
- **join_type** can be used in the following formats:
  - [INNER] JOIN
  - LEFT [OUTER] JOIN
  - RIGHT [OUTER] JOIN
  - FULL [OUTER] JOIN
  - LEFT [SEMI] JOIN
  - RIGHT [SEMI] JOIN
  - LEFT [ANTI] JOIN
  - RIGHT [ANTI] JOIN
  - CROSS JOIN
- **grouping_element** can be:
  - ()
  - expression
  - GROUPING SETS ((column [, ...]) [, ...])
  - CUBE (column [, ...])
  - ROLLUP (column [, ...])

Description

This statement is used to retrieve row data from zero or more tables.

Query the content of the **stu** table.

```
SELECT id,name FROM stu;
```

11.13.5.2 WITH

The WITH clause defines the naming relationship of query clauses, which can flatten nested queries or simplify subquery statements.

For example, the following query statements are equivalent:

```
SELECT name, maxprice FROM (SELECT name, MAX(price) AS maxprice FROM fruit GROUP BY name) AS x;

WITH x AS (SELECT name, MAX(price) AS maxprice FROM fruit GROUP BY name) SELECT name, maxprice
FROM x;
```

- Multiple Subqueries

  ```
  with
  t1 as(select name,max(price) as maxprice from fruit group by name),
  t2 as(select name,avg(price) as avgprice from fruit group by name)
  select t1.*,t2.* from t1 join t2 on t1.name = t2.name;
  ```

- WITH Chain Form

  ```
  WITH
  x AS (SELECT a FROM t),
  y AS (SELECT a AS b FROM x),
  z AS (SELECT b AS c FROM y)
  SELECT c FROM z;
  ```

11.13.5.3 GROUP BY

GROUP BY

GROUP BY groups the output rows of a SELECT statement into groups that contain matching values. A simple GROUP BY can contain any expression consisting of input columns, or select the sequence number of the output column by position.

The following queries are equivalent:

```
SELECT count(*), nationkey FROM customer GROUP BY 2;
SELECT count(*), nationkey FROM customer GROUP BY nationkey;
```

GROUP BY can group the output by the input column names that do not appear in the output of the SELECT statement.

Example:

```
SELECT count(*) FROM customer GROUP BY mktsegment;
GROUPING SETS
```

You can specify multiple columns for grouping. The result column that does not belong to the grouping column is set to **NULL**. Queries with complex grouping syntax (GROUPING SETS, CUBE, or ROLLUP) read the underlying data source only once, while queries using UNION ALL read the underlying data three times. This is why queries that use UNION ALL can produce inconsistent results when the data source is not deterministic.

```
-- Create a shipping table:
create table shipping(origin_state varchar(25),origin_zip integer,destination_state
varchar(25) ,destination_zip integer,package_weight integer);

--Insert data.
insert into shipping values ('California',94131,'New Jersey',8648,13),
('California',94131,'New Jersey',8540,42),
('California',90210,'Connecticut',6927,1337),
('California',94131,'Colorado',80302,5),
('New York',10002,'New Jersey',8540,3),
('New Jersey',7081,'Connecticut',6708,225);

-- Query the grouping sets.
SELECT
    origin_state,
    origin_zip,
```



```
    destination_state,
    sum( package_weight )
FROM shipping
GROUP BY GROUPING SETS (
        ( origin_state ),
    ( origin_state, origin_zip ),
    ( destination_state ));
--Logically, this query is equivalent to the union all of multiple group queries.
SELECT origin_state, NULL,NULL,sum( package_weight ) FROM shipping GROUP BY origin_state UNION
ALL  SELECT origin_state,origin_zip,NULL,sum( package_weight ) FROM shipping GROUP BY
origin_state,origin_zip UNION ALL  SELECT NULL,NULL,destination_state,sum( package_weight ) FROM
shipping GROUP BY  destination_state;
--Result
 origin_state | origin_zip | destination_state | _col3
--------------|------------|-------------------|-------
 New Jersey   |       NULL | NULL              |   225
 California   |      94131 | NULL              |    60
 California   |       NULL | NULL              |  1397
 New York     |      10002 | NULL              |     3
 NULL         |       NULL | New Jersey        |    58
 NULL         |       NULL | Connecticut       |  1562
 California   |      90210 | NULL              |  1337
 New York     |       NULL | NULL              |     3
 NULL         |       NULL | Colorado          |     5
 New Jersey   |       7081 | NULL              |   225
(10 rows)
```

CUBE

Generate all possible groups for given columns. For example, the possible groups of (**origin_state**, **destination_state**) are **(origin_state, destination_state)**, **(origin_state)**, **(destination_state)**, and **()**.

```
SELECT
    origin_state,
    destination_state,
    sum( package_weight )
FROM
    shipping
GROUP BY
    CUBE ( origin_state, destination_state );
-- Equivalent to:
SELECT
origin_state,
destination_state,
sum( package_weight )
FROM
    shipping
GROUP BY
    GROUPING SETS (
        ( origin_state, destination_state ),
        ( origin_state ),
    ( destination_state ),
    ());
```

ROLLUP

Generates partial possible subtotals for a given set of columns.

```
SELECT
    origin_state,
    origin_zip,
    sum( package_weight )
FROM
    shipping
GROUP BY
    ROLLUP ( origin_state, origin_zip );
-- Equivalent to:
SELECT
```



```
origin_state,
origin_zip,
sum( package_weight )
FROM
   shipping
GROUP BY
   GROUPING SETS ((origin_state,origin_zip ),( origin_state ),());
```

NOTE

Currently, GROUP BY does not support column aliases. For example:

select count(userid) as num ,dept as aaa from salary group by aaa having sum(sal)>2000;

The following error is reported:

Query 20210630_084610_00018_wc8n9@default@HetuEngine failed: line 1:63: Column 'aaa' cannot be resolved

11.13.5.4 HAVING

HAVING

HAVING is used with aggregate functions and GROUP BY to control which groups are selected.

HAVING filters out groups that do not meet specified conditions after grouping and aggregation calculation.

Example:

```
SELECT count(*), mktsegment, nationkey,
CAST(sum(acctbal) AS bigint) AS totalbal
FROM customer
GROUP BY mktsegment, nationkey
HAVING sum(acctbal) > 5700000
ORDER BY totalbal DESC;
```

11.13.5.5 UNION | INTERSECT | EXCEPT

UNION, INTERSECT, and EXCEPT are collection operations. All of them are used to merge the result sets of multiple SELECT statements into a single result set.

UNION

UNION merges all rows in the result set of the first query with rows in the result set of the second query.

query UNION [ALL | DISTINCT] query

ALL and DISTINCT indicate whether duplicate rows are returned. ALL returns all rows. DISTINCT returns only one row. If this parameter is not specified, the default value DISTINCT is used.

INTERSECT

query INTERSECT [DISTINCT] query

INTERSECT returns only the rows that intersect the results of the first and second queries. The following is an example of one of the simplest INTERSECT clauses. It

selects values **13** and **42** and merges this result set with the second query that selects value 13. Because **42** is only in the result set of the first query, it is not included in the final result:

```
SELECT * FROM (VALUES 13,42) INTERSECT SELECT 13;
 _col0 -------
    13
 (1 row)
```

EXCEPT

query EXCEPT [DISTINCT] query

EXCEPT returns rows that are in the first query result and not in the second query result.

```
SELECT * FROM (VALUES 13, 42) EXCEPT SELECT 13;
 _col0
-------
   42
(1 row)
```

NOTE

Currently, the having clause does not support column aliases. For example:

select count(userid) as num ,dept as aaa from salary group by dept having aaa='d1';

The following error is reported:

Query 20210630_085136_00024_wc8n9@default@HetuEngine failed: line 1:75: Column 'aaa' cannot be resolved

11.13.5.6 ORDER BY

ORDER BY

The ORDER BY clause is used to sort the result set by one or more output expressions.

ORDER BY expression [ASC | DESC] [NULLS { FIRST | LAST }] [, ...]

Each expression can consist of output columns or you can select the sequence number of an output column by position.

The ORDER BY clause is executed after the GROUP BY or HAVING clause and before the OFFSET, LIMIT, or FETCH FIRST clause.

NOTICE

According to the SQL specification, the ORDER BY clause affects only the row order of the query result that contains this clause. HetuEngine complies with this specification and deletes redundant usage of the clause to avoid negative impact on performance.

For example, when an INSERT statement is executed, the ORDER BY clause does not affect the inserted data. It is a redundant operation and adversely affects the overall performance of the INSERT statement. Therefore, HetuEngine skips this ORDER BY clause.

- ORDER BY applies only to the SELECT clause.
  ```
  INSERT INTO some_table
  SELECT * FROM another_table
  ORDER BY field;
  ```
- The example of ORDER BY redundancy is nested query, which does not affect the result of the entire statement.
  ```
  SELECT *
  FROM some_table
  JOIN (SELECT * FROM another_table ORDER BY field) u
  ON some_table.key = u.key;
  ```

11.13.5.7 OFFSET

OFFSET

OFFSET is used to discard the first several rows of data in the result set.

OFFSET count [ROW | ROWS]

If ORDER BY exists, OFFSET applies to the sorted result set. OFFSET discards the first several rows of data and retains the sorted data set.

```
SELECT name FROM fruit ORDER BY name OFFSET 3;
name
------------
peach
pear
watermelon
(3 rows)
```

Otherwise, if ORDER BY is not used, the discarded row may be any row. If the number of rows specified by OFFSET is greater than or equal to the size of the result set, the returned result is null.

11.13.5.8 LIMIT | FETCH FIRST

Both LIMIT and FETCH FIRST can limit the number of rows in the result set. **LIMIT** and **OFFSET** can be used together for pagination query.

LIMIT

LIMIT { count | ALL }

The following query limits the number of returned rows to 5:

```
SELECT * FROM fruit LIMIT 5;
```

LIMIT ALL has the same function as omitting LIMIT.

FETCH FIRST

FETCH { FIRST | NEXT } [count] { ROW | ROWS } { ONLY | WITH TIES

FETCH FIRST supports the FIRST or NEXT keywords and the ROW or ROWS keywords. These keywords are equivalent and do not affect the execution of queries.

- If FETCH FIRST is not specified, the default value **1** is used.
  ```
  SELECT orderdate FROM orders FETCH FIRST ROW ONLY;
   orderdate
  ```



```
--------------
 2020-11-11

SELECT *  FROM new_orders FETCH FIRST 2 ROW ONLY;
   orderkey   | orderstatus | totalprice |  orderdate
--------------|-------------|------------|--------------
 202011181113 | online      |     9527.0 | 2020-11-11
 202011181114 | online      |      666.0 | 2020-11-11
(2 rows)
```

- If OFFSET is used, LIMIT or FETCH FIRST applies to the result set after OFFSET:

```
SELECT * FROM (VALUES 5, 2, 4, 1, 3) t(x) ORDER BY x OFFSET 2  FETCH FIRST ROW ONLY;
 x
---
 3
(1 row)
```

- For the FETCH FIRST clause, the ONLY or WITH TIES parameter controls which rows are contained in the result set.

  If the ONLY parameter is specified, the result set is limited to the first several rows that contain the number of parameters.

  If WITH TIES is specified, the ORDER BY clause must be used. The result set contains the basic result set of the first several rows that meet the conditions and additional rows. These extra return rows are the same as the parameters of ORDER BY in the last row of the basic result set:

```
CREATE TABLE nation (name varchar,  regionkey integer);

insert into nation values ('ETHIOPIA',0),('MOROCCO',0),('ETHIOPIA',2),('KENYA',2),('ALGERIA',0),
('MOZAMBIQUE',0);

--Return all records whose regionkey is the same as the first record.
SELECT name, regionkey FROM nation ORDER BY regionkey FETCH FIRST ROW WITH TIES;
    name    | regionkey
------------|-----------
 ALGERIA    |         0
 ETHIOPIA   |         0
 MOZAMBIQUE |         0
 MOROCCO    |         0
(4 rows)
```

11.13.5.9 TABLESAMPLE

There are two sampling methods: BERNOULLI and SYSTEM.

Neither of these sampling methods allows you to limit the number of rows returned by the result set.

BERNOULLI

Each row is selected to the sample table based on the specified sampling rate. When the Bernoulli method is used to sample a table, all physical blocks of the table are scanned and some rows are skipped (based on the comparison between the sampling percentage and the random value calculated at run time). The probability that the result contains one row is irrelevant to any other rows. This does not reduce the time required to read the sample table from disk. If the sampling output is further processed, the total query time may be affected.

```
SELECT * FROM users TABLESAMPLE BERNOULLI (50);
```


SYSTEM

This sampling method divides a table into logical segments of data and samples the table based on the granularity. This sampling method either selects all rows from a particular data segment or skips it (based on a comparison between the sampling percentage and a random value calculated at run time). The selection of rows in the system sampling depends on the connector used. For example, if the Hive data source is used, this depends on the data layout on HDFS. This sampling method cannot ensure an independent sampling probability.

```
SELECT * FROM users TABLESAMPLE SYSTEM (75);
```

11.13.5.10 UNNEST

UNNEST can expand ARRAY or MAP to form a relation.

ARRAYS is expanded into a single column, and MAP is expanded into two columns (**key**, **value**).

UNNEST can also be used together with multiple parameters, which will be expanded into multiple columns with the same number of rows as the maximum base parameter (other columns are filled with nulls).

UNNEST can choose to use the WITH ORDINALITY clause, in which case an additional ORDINALITY column is added at the end.

UNNEST is usually used together with JOIN, which can reference columns in the left relationship of JOIN.

- Using a Separate Column
  ```
  SELECT student, score FROM tests CROSS JOIN UNNEST(scores) AS t (score);
  ```
- Using Multiple Columns
  ```
  SELECT numbers, animals, n, a
  FROM (
  VALUES
  (ARRAY[2, 5], ARRAY['dog', 'cat', 'bird']),
  (ARRAY[7, 8, 9], ARRAY['cow', 'pig'])
  ) AS x (numbers, animals)
  CROSS JOIN UNNEST(numbers, animals) AS t (n, a);
  ```

11.13.5.11 JOINS

Data of multiple relations can be combined.

HetuEngine supports the following types of JOIN: CROSS JOIN, INNER JOIN, OUTER JOIN (LEFT JOIN, RIGHT JOIN, FULL JOIN), SEMIN JOIN, and ANTI JOIN.

CROSS JOIN

CROSS JOIN returns the Cartesian product of two relationships. You can specify multiple relations using the CROSS JOIN syntax or the FROM subclause.

The following queries are equivalent:

```
SELECT * FROM nation CROSS JOIN region;
SELECT * FROM nation, region;
```


INNER JOIN

Rows can be returned only when two tables contain at least one piece of matched data, which is equivalent to JOIN. The clause can also be converted to an equivalent WHERE statement as follows:

```
SELECT * FROM nation (INNER) JOIN region ON nation.name=region.name;
SELECT * FROM nation ,region WHERE nation.name=region.name;
```

OUTER JOIN

OUTER JOIN returns all rows, both matched and unmatched, in both tables. It subdivides further into:

- Left outer join: LEFT JOIN or LEFT OUTER JOIN. This clause returns all rows from the left table (nation) and matched rows from the right table (region) based on the left table. If a row in the left table is not matched in the right table, the value of the row in the right table is NULL.
- Right outer join: RIGHT JOIN or RIGHT OUTER JOIN. This clause returns all rows from the right table (region) and matched rows from the left table (nation) based on the right table. If a row in the right table is not matched in the left table, the value of the row in the left table is NULL.
- Full outer join: FULL JOIN or FULL OUTER JOIN. This clause returns matched rows as long as there are matches in either of the tables. It is equivalent to the combination of LEFT JOIN and RIGHT JOIN.

```
SELECT * FROM nation LEFT (OUTER) JOIN region ON nation.name=region.name;
SELECT * FROM nation RIGHT (OUTER) JOIN region ON nation.name=region.name;
SELECT * FROM nation FULL (OUTER) JOIN region ON nation.name=region.name;
```

LATERAL

The LATERAL keyword can be added to the FROM subquery to allow referencing of the columns provided by the FROM item.

```
SELECT name, x, y FROM nation CROSS JOIN LATERAL (SELECT name || ' :-' AS x) CROSS JOIN LATERAL
(SELECT x || ')' AS y);
```

SEMI JOIN and ANTI JOIN

When a table finds a matched record in another table, **semi-join** returns the record in the first table. Contrary to conditional join, the table on the left node returns only one record even if several matching records are found on the right node. In addition, no record in the table on the right node is returned. The semi-join usually uses IN or EXISTS as the connection condition.

anti-join is opposite to **semi-join**. That is, the records in the first table are returned only when no matching record is found in the second table. It is used when **not exists** or **not in** is used.

Other supported conditions are as follows:

- Multiple conditions in the WHERE clause
- Alias relationships
- Subscript expressions
- Dereference expressions

- Forcible conversion expressions
- Specific functions

NOTICE

Currently, multiple semi or anti join expressions are supported only when the columns in the first table are queried in the subsequent join expressions and are not related to other join expressions.

Example:

```
CREATE SCHEMA testing ;

USE testing;

CREATE TABLE table1(id int, name varchar,rank int);

INSERT INTO table1 VALUES(10,'sachin',1),(45,'rohit',2),(46,'rohit',3),(18,'virat',4),(25,'dhawan',5);

CREATE TABLE table2(serial int,name varchar);

INSERT INTO table2 VALUES(1,'sachin'),(2,'sachin'),(3,'rohit'),(4,'virat');

CREATE TABLE table3(serial int, name varchar,country varchar);

INSERT INTO table3 VALUES(1,'sachin','india'),(20,'bhuvi','india'),(45,'boult','newzealand'),
(3,'maxwell','australia'),(45,'rohit','india'),(4,'pant','india'),(10,'KL','india'),(445,'rohit','india');

CREATE TABLE table4(id int, name varchar,rank int);

INSERT INTO table4 VALUES(10,'sachin',1),(45,'rohit',2),(46,'rohit',3),(18,'virat',4),(25,'dhawan',5);

select * from table1 left semi join table2 on table1.name=table2.name where table1.name='rohit' and
table2.serial=3;
 id | name  | rank
----|-------|------
 45 | rohit |    2
 46 | rohit |    3
(2 rows)


select * from table1 left anti join table2 on table1.name=table2.name where table1.name='rohit' and
table2.serial=3;
 id |  name  | rank
----|--------|------
 10 | sachin |    1
 18 | virat  |    4
 25 | dhawan |    5
(3 rows)

select * from table1 right semi join table2 on table1.name=table2.name where table1.name='rohit' and
table2.serial=3;
 serial | name
--------|-------
      3 | rohit
(1 row)

select * from table1 right anti join table2 on table1.name=table2.name where table1.name='rohit' and
table2.serial=3;
 serial |  name
--------|--------
      1 | sachin
      2 | sachin
      4 | virat
(3 rows)
```



```
SELECT * FROM table1 t1 LEFT SEMI JOIN table2 t2 on t1.name=t2.name left semi join table3 t3 on
t1.name = t3.name left semi join table4 t4 on t1.name=t4.name;
id | name  | rank
----|-------|------
 10 | sachin |   1
 45 | rohit |    2
 46 | rohit |    3
(3 rows)
```

Qualifying Column Names

When two relations of JOIN have the same column name, the relation alias (if any) or relation name must be used for column reference.

```
SELECT nation.name, region.name FROM nation CROSS JOIN region;
SELECT n.name, r.name FROM nation AS n CROSS JOIN region AS r;
SELECT n.name, r.name FROM nation n CROSS JOIN region r;
```

11.13.5.12 Subqueries

EXISTS

The EXISTS predicate determines whether to return any row.

```
SELECT name FROM nation WHERE EXISTS (SELECT * FROM region WHERE region.regionkey =
nation.regionkey)
```

IN

It determines whether any value generated by a subquery is equal to a given expression.

The IN result complies with the standard null rule.

Only one column must be generated for a subquery.

```
SELECT name FROM nation WHERE regionkey IN (SELECT regionkey FROM region)
```

11.13.5.13 SELECT VIEW CONTENT

- Syntax

  SELECT column_name FROM view_name
- Description

  This statement is used to query view content.
- Example

  ```
  SELECT * FROM test_view;
  ```

11.13.5.14 REWRITE HINT

This statement can be used together with a SELECT statement to rewrite a query using a specified materialized view, which accelerates the execution of the query statement. A hint must be provided at the beginning of a query. Currently, the following two types of hints are supported:

- NOREWRITE

  Disables query rewrite. Format: /*+ NOREWRITE */

- REWRITE(materialized_view_name ..)

  Rewrites a query based on the known materialized view name. You can enter multiple materialized view names and separate them by spaces. A materialized view name must be fully qualified.

  Format: /*+ REWRITE(mv1 mv2 ..) */

Example

- Forcibly execute the original SQL query without rewriting the query.

```
/*+ NOREWRITE */ SELECT c1,c2 FROM table;
```

- Rewrite a query using a specified materialized view.

```
SET SESSION materialized_view_rewrite_enabled=true; -- Enable the function of query rewrite using
materialized views.
CREATE TABLE t1 (id int, c1 varchar);
INSERT INTO t1 VALUES (1,'abc'), (2,'abc2'), (3,'abc3'), (4,'abc4'), (5,'abc5'),(6,'abc6');
CREATE TABLE t2 (id1 int, c1 varchar);
INSERT INTO t2 VALUES (1,'abc'), (2,'abc2'), (3,'abc3'), (4,'abc4'), (5,'abc5'),(6,'abc6');
-- Create a materialized view for a SQL query and create materialized views for the subqueries
contained in the query.
CREATE MATERIALIZED VIEW mv.tpcds.test7 AS SELECT a.id,b.c1 FROM (SELECT id FROM t1 WHERE
id>5) as a,(SELECT id1,c1 FROM t2 WHERE id1>4) AS b WHERE a.id = b.id1;
CREATE MATERIALIZED VIEW mv.tpcds.test6a AS SELECT id FROM t1 WHERE id>5;
CREATE MATERIALIZED VIEW mv.tpcds.test6b AS SELECT id1,c1 FROM t2 WHERE id1>4;
-- Rewrite a query using the materialized view of a specified subquery.
EXPLAIN /*+ REWRITE(mv.tpcds.test6a mv.tpcds.test6b) */ SELECT a.id,b.c1 FROM (SELECT id FROM t1
WHERE id>5) AS a,(SELECT id1,c1 FROM t2 WHERE id1>4) AS b WHERE a.id = b.id1;
                                              Query Plan
-------------------------------------------------------------------------------------------------------------------
------
 Output[id, c1]
 │   Layout: [id:integer, c1:varchar]
 │   Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
 └─ RemoteExchange[GATHER]
    │   Layout: [id:integer, c1:varchar]
    │   Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
    └─ InnerJoin[("id" = "id1")][$hashvalue,
$hashvalue_22]
       │   Layout: [id:integer, c1:varchar]
       │   Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
       │   Distribution: PARTITIONED
       ├─ RemoteExchange[REPARTITION]
[$hashvalue]
       │  │   Layout: [id:integer, $hashvalue:bigint]
       │  │   Estimates: {rows: ? (?), cpu: ?, memory: 0B,
network: ?}
       │  └─ ScanProject[table = hive:tpcds:test6a]
       │         Layout: [id:integer, $hashvalue_21:bigint]
       │         Estimates: {rows: ? (?), cpu: ?, memory: 0B, network: 0B}/{rows: ? (?), cpu: ?, memory: 0B,
network: 0B}
       │         $hashvalue_21 := combine_hash(BIGINT 0, COALESCE($operator$hash_code(id), BIGINT
0))
       │         id := id:int:0:REGULAR
       └─ LocalExchange[HASH][$hashvalue_22]
("id1")
          │   Layout: [id1:integer, c1:varchar, $hashvalue_22:bigint]
          │   Estimates: {rows: ? (?), cpu: ?, memory: 0B,
network: ?}
          └─ RemoteExchange[REPARTITION]
[$hashvalue_23]
             │   Layout: [id1:integer, c1:varchar, $hashvalue_23:bigint]
             │   Estimates: {rows: ? (?), cpu: ?, memory: 0B,
network: ?}
             └─ ScanProject[table = hive:tpcds:test6b]
                    Layout: [id1:integer, c1:varchar, $hashvalue_24:bigint]
                    Estimates: {rows: ? (?), cpu: ?, memory: 0B, network: 0B}/{rows: ? (?), cpu: ?, memory:
0B, network: 0B}
```

```
                $hashvalue_24 := combine_hash(BIGINT 0, COALESCE($operator$hash_code(id1),
BIGINT 0))
                id1 := id1:int:0:REGULAR
                c1 := c1:string:1:REGULAR

(1 row)

-- Rewrite a query using the materialized view of a specified SQL query.
EXPLAIN SELECT a.id,b.c1 FROM (SELECT id FROM t1 WHERE id>5) AS a,(SELECT id1,c1 FROM t2
WHERE id1>4) AS b WHERE a.id = b.id1;
                        Query Plan
------------------------------------------------------------------------
 Output[id, c1]
 │   Layout: [id:integer, c1:varchar]
 │   Estimates: {rows: ? (?), cpu: ?, memory: 0B, network: ?}
 └─ RemoteExchange[GATHER]
    │   Layout: [id:integer, c1:varchar]
    │   Estimates: {rows: ? (?), cpu: ?, memory: 0B, network: ?}
    └─ TableScan[table = hive:tpcds:test7]
       Layout: [id:integer, c1:varchar]
       Estimates: {rows: ? (?), cpu: ?, memory: 0B, network: 0B}
       id := id:int:0:REGULAR
       c1 := c1:string:1:REGULAR

(1 row)
```

11.13.6 HetuEngine SQL Functions and Operators

11.13.6.1 Logical Operators

Logical Operators

Operation	Description	Example
AND	If both values are **true**, the value is **true**.	a AND b
OR	If one of the two values is **true**, the value is **true**.	a OR b
NOT	If the value is **false**, the result is **true**.	NOT a

The following truth table reflects how AND and OR handle NULL values.

a	b	a AND b	a OR b
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	NULL	NULL	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE
FALSE	NULL	FALSE	NULL

a	b	a AND b	a OR b
NULL	TRUE	NULL	TRUE
NULL	FALSE	FALSE	NULL
NULL	NULL	NULL	NULL

The following truth table reflects how NOT handle NULL values.

value	NOT value
TRUE	FALSE
FALSE	TRUE
NULL	NULL

11.13.6.2 Comparison Functions and Operators

Comparison

Operation	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
=	Equal to
<>	Not equal to
!=	Not equal to

- Scope comparison: **between**

  between is applicable to values in a specified range, for example, *value* **BETWEEN** *min* **AND** *max*.

  Not between is used when the value is not in a specified range.

  The **null** value cannot be used in the **between** operation. The execution results of the following two operations are **Null**:

  ```
  SELECT NULL BETWEEN 2 AND 4; -- null
  SELECT 2 BETWEEN NULL AND 6; -- null
  ```

  In HetuEngine, the *value*, *min*, and *max* parameters must be of the same data type in **BETWEEN** and **NOT BETWEEN**.

  Wrong usage: **'John' between 2.3 and 35.2**

Example expression equivalent to BETWEEN:

```
SELECT 3 BETWEEN 2 AND 6; -- true
SELECT 3 >= 2 AND 3 <= 6; -- true
```

Example expression equivalent to NOT BETWEEN:

```
SELECT 3 NOT BETWEEN 2 AND 6; -- false
SELECT 3 < 2 OR 3 > 6; -- false
```

- **IS NULL** and **IS NOT NULL**

  They are used to determine whether a value is empty. All data types can be used for this determination.

  ```
  SELECT 3.0 IS NULL; -- false
  ```

- **IS DISTINCT FROM** and **IS NOT DISTINCT FROM**

  This is a special usage. In HetuEngine SQL statements, NULL indicates an unknown value. All comparisons related to NULL also produce NULL results. **IS DISTINCT FROM** and **IS NOT DISTINCT FROM** can take a null value as a known value and return **true** or **false** (even if the expression contains a null value).

  Example:

  ```
  --Create a table.
  create table dis_tab(col int);
  --Insert data.
  insert into dis_tab values (2),(3),(5),(null);
  --Query:
  select col from dis_tab where col is distinct from null;
   col
  ----
   2
   3
   5
  (3 rows)
  ```

  The following truth table demonstrates how IS DISTINCT FROM and IS NOT DISTINCT FROM process common data and NULL values.

a	b	a = b	a <> b	a DISTINCT b	a NOT DISTINCT b
1	1	TRUE	FALSE	FALSE	TRUE
1	2	FALSE	TRUE	TRUE	FALSE
1	NULL	NULL	NULL	TRUE	FALSE
NULL	NULL	NULL	NULL	FALSE	TRUE

- **GREATEST** and **LEAST**

  The two functions are not standard SQL functions. They are typical extensions. The parameter cannot contain **null** values.

  – greatest(value1, value2, ..., valueN)

    Returns the provided maximum value.

  – least(value1, value2, ..., valueN) → [same as input]

    Returns the provided minimum value.

- Batch comparison: **ALL**, **ANY**, and **SOME**

Quantifiers ALL, ANY, and SOME can be used together with comparison operators in the following ways:

```
expression operator quantifier ( subquery )
```

The meanings of some combinations of quantifiers and comparison operators are as follows. ANY and SOME have the same meaning. You can replace ANY with SOME when using the expressions in the following table.

Expression	Definition
A = ALL (...)	Returns **true** when A is equal to all values.
A <> ALL (...)	Returns **true** when A is not equal to any value.
A < ALL (...)	Returns **true** when A is less than the minimum value.
A = ANY (...)	Returns **true** when A is the same as any value, which is equivalent to A IN (...).
A <> ANY (...)	Returns **true** when A is different from any value.
A < ANY (...)	Returns **true** when A is less than the maximum value.

Example:

```
SELECT 'hello' = ANY (VALUES 'hello', 'world'); -- true
SELECT 21 < ALL (VALUES 19, 20, 21); -- false
SELECT 42 >= SOME (SELECT 41 UNION ALL SELECT 42 UNION ALL SELECT 43);-- true
```

11.13.6.3 Condition Expression

CASE

Standard SQL CASE expressions have two modes.

- In simple mode, search for each value of the expression from left to right until the same expression is found.

  CASE expression

  WHEN value THEN result

  [WHEN ...]

  [ELSE result]

  END

  Returns the result that matches the value. If no value is matched, the result of the **ELSE** clause is returned. If there is no **ELSE** clause, **null** is returned.
  Example:

```
select a,
case a
 when 1 then 'one'
```



```
when 2 then 'two'
 else 'many' end from
 (values (1),(2),(3),(4)) as t(a);
 a | _col1
---|-------
 1 | one
 2 | two
 3 | many
 4 | many
(4 rows)
```

- In search mode, the system checks the Boolean value of each condition from left to right until the value is true and returns the matching result.

  CASE

  WHEN condition THEN result

  [WHEN ...]

  [ELSE result] END

  If none of the conditions is met, the result of the **ELSE** clause is returned. If there is no **ELSE** clause, null is returned. Example:

```
select a,b,
case
when a=1 then 'one'
when b=2 then 'tow'
else 'many' end from (values (1,2),(3,4),(1,3),(4,2)) as t(a,b);
 a | b | _col2
---|---|-------
 1 | 2 | one
 3 | 4 | many
 1 | 3 | one
 4 | 2 | tow
(4 rows)
```

IF

The IF function is a language structure. It has the same function as the following CASE expression:

CASE

WHEN condition THEN true_value

[ELSE false_value] END

- if(condition, true_value)

  If *condition* is **true**, *true_value* is returned. Otherwise, **null** is returned and *true_value* is not calculated.

```
select if(a=1,8) from (values (1),(1),(2)) as t(a); -- 8 8 NULL
select if(a=1,'value') from (values (1),(1),(2)) as t(a); -- value value NULL
```

- if(condition, true_value, false_value)

  If *condition* is **true**, *true_value* is returned. Otherwise, *false_value* is returned.

```
select if(a=1,'on','off') from (values (1),(1),(2)) as t(a);
_col0
-------
 on
 on
 off
(3 rows)
```


COALESCE

coalesce(value[, ...])

Returns the first non-null value in the parameter list. Similar to CASE expressions, parameters are calculated only when necessary.

It is similar to the nvl function of MySQL and is often used to convert a null value to 0 or ' ' (null character).

```
select coalesce(a,0) from (values (2),(3),(null)) as t(a); -- 2 3 0
```

NULLIF

- nullif(value1, value2)

  If *value1* is equal to *value2*, **null** is returned. Otherwise, *value1* is returned.

  ```
  select nullif(a,b) from (values (1,1),(1,2)) as t(a,b); --
   _col0
  -------
    NULL
       1
  (2 rows)
  ```

- ZEROIFNULL(value)

  If the value is **null**, **0** is returned. Otherwise, the original value is returned. Currently, the varchar type is also supported.

  ```
  select zeroifnull(a),zeroifnull(b),zeroifnull(c) from (values (null,13.11,bigint '157'),(88,null,bigint '188'),
  (55,14.11,null)) as t(a,b,c);
   _col0 | _col1 | _col2
  -------|-------|-------
       0 | 13.11 |   157
      88 |  0.00 |   188
      55 | 14.11 |     0
  (3 rows)
  ```

- NVL(value1,value2)

  If *value1* is **null**, *value2* is returned. Otherwise, *value1* is returned.

  ```
  select nvl(NULL,3);  -- 3
  select nvl(2,3);    --2
  ```

- ISNULL(value)

  If *value1* is **null**, **true** is returned. Otherwise, **false** is returned.

  ```
  Create table nulltest(col1 int,col2 int);
  insert into nulltest values(null,3);
  select isnull(col1),isnull(col2) from nulltest;
   _col0 | _col1
  -------|-------
   true  | false
  (1 row)
  ```

- ISNOTNULL(value)

  If *value1* is **null**, **false** is returned. Otherwise, **true** is returned.

  ```
  select isnotnull(col1),isnotnull(col2) from nulltest;
   _col0 | _col1
  -------|-------
   false | true
  (1 row)
  ```

TRY

Evaluates an expression. If an error occurs, **Null** is returned. It is similar to **try catch** in the programming language. The **TRY** function is generally used together

with **COALESCE**. **COALESCE** can convert an abnormal null value to **0** or **null**. The following situations will be captured by the **TRY** function:

- The denominator is 0.
- The cast operation or function input parameter is incorrect.
- The number exceeds the defined length.

This method is not recommended. Specify the preceding exceptions and preprocess data.

Example:

Assume that the **origin_zip** field in the following table contains invalid data:

```
--Create a table.
create  table shipping (origin_state varchar,origin_zip varchar,packages int ,total_cost int);

--Insert data.
insert into shipping
values
('California','94131',25,100),
('California','P332a',5,72),
('California','94025',0,155),
('New Jersey','08544',225,490);

--Query data.
SELECT * FROM shipping;
 origin_state | origin_zip | packages | total_cost
 --------------+------------+----------+------------
 California   |      94131 |       25 |        100
 California   |      P332a |        5 |         72
 California   |      94025 |        0 |        155
 New Jersey   |      08544 |      225 |        490
(4 rows)
```

The query fails when **TRY** is not used:

```
SELECT CAST(origin_zip AS BIGINT) FROM shipping;
Query failed: Cannot cast 'P332a' to BIGINT
```

When **TRY** is used, **null** is returned:

```
SELECT TRY(CAST(origin_zip AS BIGINT)) FROM shipping;
 origin_zip
 ------------
     94131
     NULL
     94025
     08544
 (4 rows)
```

The query fails when **TRY** is not used:

```
SELECT total_cost/packages AS per_package FROM shipping;
Query failed: Division by zero
```

The default values are returned when **TRY** and **COALESCE** are used.

```
SELECT COALESCE(TRY(total_cost/packages),0) AS per_package FROM shipping;
 per_package
 -------------
    4
    14
    0
    19
 (4 rows)
```


11.13.6.4 Lambda Expression

The Lambda expression can be represented by ->.

```
x->x+1
(x,y)->x+y
x->regexp_like(x,'a+')
x->x[1]/x[2]
x->IF(x>0,x,-x)
x->COALESCE(x,0)
x->CAST(xASJSON)
x->x+TRY(1/0)
```

Most SQL expressions can be used in the Lambda function body except in the following scenarios:

- Subqueries are not supported.
  ```
  x -> 2 + (SELECT 3)
  ```
- Aggregate functions are not supported.
  ```
  x -> max(y)
  ```

Examples

- Use the **transform()** function to obtain the square of array elements.
  ```
  SELECT numbers, transform(numbers, n -> n * n) as squared_numbers FROM (VALUES (ARRAY[1, 2]), (ARRAY[3, 4]),(ARRAY[5, 6, 7])) AS t(numbers);
    numbers  | squared_numbers
  -----------|-----------------
   [1, 2]    | [1, 4]
   [3, 4]    | [9, 16]
   [5, 6, 7] | [25, 36, 49]
  (3 rows)
  ```
- Use the **transform()** function to convert array elements into strings. If the conversion fails, the array elements will be converted into NULL to avoid errors.
  ```
  SELECT transform(prices, n -> TRY_CAST(n AS VARCHAR) || '$') as price_tags FROM (VALUES (ARRAY[100, 200]),(ARRAY[30, 4])) AS t(prices);
    price_tags
  --------------
   [100$, 200$]
   [30$, 4$]
  (2 rows)
  ```
- When an operation is performed on an array element, other columns can also be used in the operation. For example, use **transform()** to calculate the linear equation **f(x) =ax + b**.
  ```
  SELECT xvalues, a, b, transform(xvalues, x -> a * x + b) as linear_function_values FROM (VALUES (ARRAY[1, 2], 10, 5), (ARRAY[3, 4], 4, 2)) AS t(xvalues, a, b);
   xvalues | a  | b | linear_function_values
  ---------|----|---|------------------------
   [1, 2]  | 10 | 5 | [15, 25]
   [3, 4]  |  4 | 2 | [14, 18]
  (2 rows)
  ```
- Use **any_match()** to search for an array where at least one element is greater than 100.
  ```
  SELECT numbers FROM (VALUES (ARRAY[1,NULL,3]), (ARRAY[10,200,30]), (ARRAY[100,20,300])) AS t(numbers) WHERE any_match(numbers, n ->  COALESCE(n, 0) > 100);
     numbers
  ----------------
   [10, 200, 30]
   [100, 20, 300]
  (2 rows)
  ```
- Use **regexp_replace()** to capitalize the first letter.


```
SELECT regexp_replace('once upon a time ...', '^(\w)(\w*)(\s+.*)$',x -> upper(x[1]) || x[2] || x[3]); --
Once upon a time ...
```

- Use the Lambda expression in the aggregate function. For example, use **reduce_agg()** to calculate the sum of elements by column.

```
SELECT reduce_agg(value, 0, (a, b) -> a + b, (a, b) -> a + b) sum_values FROM (VALUES (1), (2), (3),
(4), (5)) AS t(value);
 sum_values
------------
         15
(1 row)
```

11.13.6.5 Conversion Functions

cast Conversion Function

HetuEngine implicitly converts numeric and character values to the correct type. HetuEngine does not convert between character and numeric types. For example, if a query expects a value of the varchar type, HetuEngine does not automatically convert a value of the bigint type to a value of the varchar type.

Values can be explicitly converted to the specified type, if necessary.

- cast(value AS type) → type

  Explicitly converts the type of a value. You can convert a value of the varchar type to the numeric type, or vice versa.

```
select cast('186' as int );
select cast(186 as varchar);
```

- try_cast(value AS type) → type

  It is similar to **cast()**. The difference is that null is returned if the conversion fails.

```
select try_cast(1860 as tinyint);
 _col0
-------
  NULL
(1 row)
```

NOTE

When a number overflows or a null value is converted, **null** is returned. However, when the conversion fails, an error is reported.

Example: select try_cast(186 as date);

Cannot cast integer to date

Format

- format(format, args...) → varchar

  Description: Formats a string in the format specified by the format string and returns the formatted string.

```
SELECT format('%s%%',123);-- '123%'
SELECT format('%.5f',pi());-- '3.14159'
SELECT format('%03d',8);-- '008'
SELECT format('%,.2f',1234567.89);-- '1,234,567.89'
SELECT format('%-7s,%7s','hello','world');-- 'hello  ,  world'
SELECT format('%2$s %3$s %1$s','a','b','c');-- 'b c a'
SELECT format('%1$tA, %1$tB %1$te, %1$tY',date'2006-07-04');-- 'Tuesday, July 4, 2006
```

- format_number(number) → varchar

Description: Returns a string formatted by unit symbol.

```
SELECT format_number(123456); -- '123K'
SELECT format_number(1000000); -- '1M'
```

Data Size

The **parse_presto_data_size** function supports the following units:

Unit	Description	Value
B	Bytes	1
kB	Kilobytes	1024
MB	Megabytes	1024^2
GB	Gigabytes	1024^3
TB	Terabytes	1024^4
PB	Petabytes	1024^5
EB	Exabytes	1024^6
ZB	Zettabytes	1024^7
YB	Yottabytes	1024^8

parse_presto_data_size(string) → decimal(38)

Convert formatted values with a unit to a number. The value can be a decimal.

```
SELECT parse_presto_data_size('1B'); -- 1
SELECT parse_presto_data_size('1kB'); -- 1024
SELECT parse_presto_data_size('1MB'); -- 1048576
SELECT parse_presto_data_size('2.3MB'); -- 2411724
```

Others

typeof(expr) → varchar

Returns the data type name of an expression.

```
SELECT typeof(123);-- integer
SELECT typeof('cat');-- varchar(3)
SELECT typeof(cos(2)+1.5);-- double
```

11.13.6.6 Mathematical Functions and Operators

Mathematical Operator

Operator	Description
+	Add
-	Deduct

Operator	Description
*	Multiple
/	Divide
%	Remainder

Mathematical Functions

- abs(x) → [same as input]
  Returns the absolute value of *x*.
  ```
  SELECT abs(-17.4);-- 17.4
  ```
- bin(bigint x) -> string
  Returns *x* in binary format.
  ```
  select bin(5); --101
  ```
- bround(double x) -> double
  Banker's rounding:
  - 1 to 4: rounding down
  - 6 to 9: rounding up
  - The number before 5 is even: rounding down
  - The number before 5 is odd: rounding up

  ```
  select bround(3.5); -- 4.0
  select bround(2.5); -- 2.0
  select bround(3.4); -- 3.0
  ```
- bround(double x, int y) -> double
  Banker's rounding with *y* decimal places reserved.
  ```
  select bround(8.35,1); --8.4
  select bround(8.355,2); --8.36
  ```
- ceil(x) → [same as input]
  Same as **ceiling()**
  ```
  SELECT ceil(-42.8); -- -42
  ```
  ceiling(x) → [same as input]
  Returns the rounded-up value of *x*.
  ```
  SELECT ceiling(-42.8); -- -42
  ```
- conv(bigint num, int from_base, int to_base)
- conv(string num, int from_base, int to_base)
  Converts **num**, for example, from decimal to binary.
  ```
  select conv('123',10,2); -- 1111011
  ```
- rand() → double
  Returns a random decimal number between 0 and 1.
  ```
  select rand();--  0.049510824616263105
  ```
- cbrt(x) → double
  Returns the cube root of *x*.
  ```
  SELECT cbrt(27.0); -- 3
  ```


- e() → double

  Returns the Euler constant.

  ```
  select e();-- 2.718281828459045
  ```

- exp(x) → double

  Returns the value of *e* raised to the power of *x*.

  ```
  select exp(1);--2.718281828459045
  ```

- factorial(int x) -> bigint

  Returns the factorial of *x*. The value range of *x* is [0, 20].

  ```
  select factorial(4); --24
  ```

- floor(x) → [same as input]

  Returns the nearest integer rounded off from *x*.

  ```
  SELECT floor(-42.8);-- -43
  ```

- from_base(string, radix) → bigint

  Converts a specified number system to bigint. For example, converts the ternary number 200 to a decimal number.

  ```
  select from_base('200',3);--18
  ```

- hex(bigint|string|binary x) -> string

  Returns a hexadecimal number as a string if *x* is of the int or binary type. If *x* is a string, converts each character of the string to a hexadecimal representation and returns a string.

  ```
  select hex(68); -- 44
  select hex('AE'); -- 4145
  ```

- to_base(*x*, *radix*) → varchar

  Converts an integer into a character string in the radix system. For example, converts the decimal number 18 to a ternary number.

  ```
  select to_base(18,3);-- 200
  ```

- ln(x) → double

  Returns the natural logarithm of *x*.

  ```
  select ln(10);--2.302585092994046
  select ln(e());--1.0
  ```

- log2(x) → double

  Returns the logarithm of *x* to base 2.

  ```
  select log2(4);-- 2.0
  ```

- log10(x) → double

  Returns the logarithm of *x* to base 10.

  ```
  select log10(1000);-- 3.0
  ```

- log(b, x) → double

  Returns the logarithm of *x* to base *b*.

  ```
  select log(3,81); -- 4.0
  ```

- mod(n, m) → [same as input]

  Returns the modulus of *n* divided by *m*.

  ```
  select mod(40,7) ;-- 5
  select mod(-40,7);  -- -5
  ```

- pi() → double

  Returns pi.


```
select pi();--3.141592653589793
```

- pmod(int x,int y) -> int
- pmod(double x,double y) -> double

  Returns the positive value of the remainder after division of **x** by **y**.

```
select pmod(8,3); --2
Select pmod(8.35,2.0); --0.35
```

- pow(x, p) → double

  Same as **power()**.

```
select pow(3.2,3);-- 32.76800000000001
```

- power(x,p)

  Returns the value of *x* raised to the power of *p*.

```
select power(3.2,3);-- 32.76800000000001
```

- radians(x) → double

  Converts the angle *x* to a radian.

```
 select radians(57.29577951308232);-- 1.0
```

- degrees(x) → double

  Converts an angle *x* (represented by a radian) into an angle.

```
select degrees(1);-- 57.29577951308232
```

- round(*x*) → [same as input]

  Return the integer that is rounded to the nearest integer of *x*.

```
select round(8.57);-- 9
```

- round(*x*, *d*) → [same as input]

  x is rounded off to *d* decimal places.

```
select round(8.57,1);-- 8.60
```

- shiftleft(tinyint|smallint|int x, int y) -> int
- shiftleft(bigint x, int y) -> bigint

  Returns the value of *x* shifted leftwards by *y* positions.

```
select shiftleft(8,2);--32
```

- shiftright(tinyint|smallint|int a, int b) -> int
- shiftright(bigint a, int b) -> bigint

  Returns the value of *x* shifted rightwards by *y* positions.

```
select shiftright(8,2);--2
```

- shiftrightunsigned(tinyint|smallint|int x, int y) -> int
- shiftrightunsigned(bigint x, int y) -> bigint

  Shifts to the right by bit without symbols, and returns the value of *x* shifted rightwards by *y* positions. Returns an int if *x* is tinyint, smallint, or int. Returns a bigint if *x* is bigint.

```
select shiftrightunsigned(8,3); -- 1
```

- sign(*x*) → [same as input]

  Returns the symbol function of x.

  - If *x* is equal to **0**, **0** is returned.
  - If *x* is less than **0**, the value **–1** is returned.
  - If *x* is greater than **0**, **1** is returned.


```
select sign(-32.133);-- -1
select sign(32.133); -- 1
select sign(0);--0
```

For parameters of the double type:

- If the parameter is NaN, **NaN** is returned.
- If the parameter is +∞, **1** is returned.
- If the parameter is -∞, **-1** is returned.

```
select sign(NaN());--NaN
select sign(Infinity());-- 1.0
select  sign(-infinity());-- -1.0
```

- sqrt(*x*) → double

  Returns the square root of *x*.

  ```
  select sqrt(100); -- 10.0
  ```

- truncate(*number,num_digits*)
  - *Number* indicates the number to be truncated, and *Num_digits* indicates the decimal places retained.
  - The default value of *Num_digits* is **0**.
  - The **truncate()** function does not round off the result.

  ```
  select truncate(10.526); -- 10
  select truncate(10.526,2); --  10.520
  ```

- trunc(*number,num_digits*)

  See truncate(*number,num_digits*).

- unhex(string x) -> binary

  Returns the reciprocal of a hexadecimal number.

  ```
  select unhex('123'); --^A#
  ```

- width_bucket(*x*, *bound1*, *bound2*, *n*) → bigint

  Returns the number of containers x in the equi-width histogram with the specified **bound1** and **bound2** boundaries and *n* buckets.

  ```
  select value,width_bucket(value,1,5000,10) from (values (1),(100),(500),(1000),(2000),(2500),(3000),(4000),(4500),(5000),(8000)) as t(value);
  value | _col1
  -------|-------
      1 |     1
    100 |     1
    500 |     1
   1000 |     2
   2000 |     4
   2500 |     5
   3000 |     6
   4000 |     8
   4500 |     9
   5000 |    11
   8000 |    11
  (11 rows)
  ```

- width_bucket(*x*, *bins*) → bigint

  Returns the number of bins of x based on the bin specified by the array **bin**. The **bins** parameter must be a double-precision array and is assumed to be in ascending order.

  ```
  select width_bucket(x,array [1.00,2.89,3.33,4.56,5.87,15.44,20.78,30.77]) from (values (3),(4)) as t(x);
   _col0
  -------
      2
      3
  (2 rows)
  ```


- quotient(BIGINT numerator, BIGINT denominator) → bigint

  Returns the value of the left number divided by the right number. Part of the decimal part is discarded.

  ```
  select quotient(25,4);-- 6
  ```

Random

- rand() → double

  Same as **random()**

- random() → double

  Returns a pseudo-random value in the range of 0.0 <= x < 1.0.

  ```
  select random();-- 0.021847965885988363
  select random();-- 0.5894438037549372
  ```

- random(*n*) → [same as input]

  Returns a pseudo-random number between 0 and n (excluding n).

  ```
  select random(5);-- 2
  ```

NOTICE

random(n) contains the following data types: tinyint, bigint, smallint and integer.

Statistical Function

The binomial distribution confidence interval has multiple calculation formulas, and the most common one is ["normal interval"]. However, it is applicable only to a case in which there are a relatively large quantity of samples (np > 5 and n(1-p) > 5). For a small sample, the accuracy is poor. To solve this problem, the Wilson Score Interval is used.

$$\left(\hat{p}+\frac{z_{\alpha/2}^2}{2n}\pm z_{\alpha/2}\sqrt{[\hat{p}(1-\hat{p})+z_{\alpha/2}^2/4n]/n}\right)/(1+z_{\alpha/2}^2/n).$$

z —— normal distribution, average value + z x standard deviation confidence. z = 1.96, confidence level: 95%

Take, for example, the collecting of positive rate. **pos** indicates the number of positive reviews; **n** indicates the total number of reviews; and **phat** indicates the positive review rate.

z = 1.96

phat= 1.0* pos/n

z1=phat + z * z/(2 * n)

z2 = $z*\sqrt{\text{paht}(1-\text{phat})/n+z^2/(4*n^2)}$

m = (1 + z * z/n)

Lower limit **(z1-z2)/m**, upper limit **(z1+z2)/m**

- wilson_interval_lower(successes, trials, z) → double

  Returns the lower bound of the Wilson score interval for the Bernoulli test process. The confidence value is specified by the z-score **z**.

  ```
  select wilson_interval_lower(1, 5, 1.96);-- 0.036223160969787456
  ```

- wilson_interval_upper(successes, trials, z) → double

  Returns the upper bound of the Wilson score interval for the Bernoulli test process. The confidence value is specified by the z-score **z**.

  ```
  select wilson_interval_upper(1, 5, 1.96);--  0.6244717358814612
  ```

- cosine_similarity(*x*, *y*) → double

  Returns the cosine similarity between sparse vectors *x* and *y*.

  ```
  SELECT cosine_similarity (MAP(ARRAY['a'],ARRAY[1.0]),MAP(ARRAY['a'],ARRAY[2.0]));-- 1.0
  ```

Cumulative Distribution Function

- beta_cdf(*a*, *b*, *v*) → double

Use the given **a** and **b** parameters to calculate the cumulative distribution function (P (N <v; a, b)) of the beta distribution. Parameters **a** and **b** must be positive real numbers, and the value **v** must be a real number. The value **v** must be within the interval [0, 1].

A cumulative distribution function formula of beta distribution is also referred to as an incomplete beta function ratio (which is usually represented by **Ix**), and corresponds to the following formula:

$$F(x) = I_x(p,q) = \frac{\int_0^x t^{p-1}(1-t)^{q-1}dt}{B(p,q)} \quad 0 \le x \le 1; p,q > 0$$

```
select beta_cdf(3,4,0.0004); --  1.278848368599041E-9
```

- inverse_beta_cdf(*a*, *b*, *p*) → double

  The inverse operation of the beta cumulative distribution function, given the **a** and **b** parameters of the cumulative probability p: P (N < n). Parameters **a** and **b** must be positive real numbers, and **p** must be within the range of [0,1].

  ```
  select inverse_beta_cdf(2, 5, 0.95) ;--0.5818034093775719
  ```

- inverse_normal_cdf(*mean*, *sd*, *p*) → double

  Given the cumulative probability (p): P (N < n) related mean and standard deviation, calculate the inverse of the normal cumulative distribution function. The average value must be a real value, and the standard deviation must be a positive real value. The probability **p** must be in the interval (0, 1).

  ```
  select inverse_normal_cdf(2, 5, 0.95);-- 10.224268134757361
  ```

- normal_cdf(*mean*, *sd*, *v*) → double

  Calculate the value of the normal distribution function based on the average value and standard deviation. P(N<v; mean,sd). The average value and **v** must be real values, and the standard deviation must be positive real values.

  ```
  select normal_cdf(2, 5, 0.95);--  0.4168338365175577
  ```


Trigonometric Function

The parameters of all trigonometric functions are expressed in radians. Refer to the unit conversion functions **degrees()** and **radians()**.

- acos(x) → double

  Calculates the arc cosine value.

  ```
  SELECT acos(-1);-- 3.14159265358979
  ```

- asin(x) → double

  Calculates the arc sine value.

  ```
  SELECT asin(0.5);-- 0.5235987755982989
  ```

- atan(x) → double

  Returns the arc tangent value of *x*.

  ```
  SELECT atan(1);-- 0.7853981633974483
  ```

- atan2(y, x) → double

  Return the arc tangent value of *y/x*.

  ```
  SELECT atan2(2,1);-- 1.1071487177940904
  ```

- cos(x) → double

  Returns the cosine value of *x*.

  ```
  SELECT cos(-3.1415927);-- -0.9999999999999989
  ```

- cosh(x) → double

  Returns the hyperbolic cosine value of *x*.

  ```
  SELECT cosh(3.1415967);-- 11.592000006553231
  ```

- sin(x) → double

  Returns the sine value of *x*.

  ```
  SELECT sin(1.57079);--  0.9999999999799858
  ```

- tan(x) → double

  Returns the tangent value of *x*.

  ```
  SELECT tan(20);-- 2.23716094422474
  ```

- tanh(x) → double

  Returns the hyperbolic tangent value of *x*.

  ```
  select tanh(3.1415927);-- 0.9962720765661324
  ```

Floating-Point Function

- infinity() → double

  Returns a constant representing positive infinity.

  ```
  select infinity();-- Infinity
  ```

- is_finite(x) → boolean

  Checks whether *x* is a finite value.

  ```
  select is_finite(infinity());-- false
  select is_finite(50000);--true
  ```

- is_infinite(x) → boolean

  Determines whether *x* is infinite.

  ```
  select is_infinite(infinity());-- true
  select is_infinite(50000);--false
  ```


- is_nan(x) → boolean

  Checks whether *x* is a non-digit character.

  ```
  -- The input value must be of the double type.
  select is_nan(null); -- NULL
  select is_nan(nan()); -- true
  select is_nan(45);-- false
  ```

- nan() → double

  Returns a constant representing a non-numeric number.

  ```
  select nan(); -- NaN
  ```

11.13.6.7 Bitwise Functions

- bit_count(x, bits) → bigint

  Calculate the number of bits set in x (regarded as an integer with a signed bit) in the complementary code representation of 2.

  ```
  SELECT bit_count(9, 64); -- 2
  SELECT bit_count(9, 8); -- 2
  SELECT bit_count(-7, 64); -- 62
  SELECT bit_count(-7, 8); -- 6
  ```

- bitwise_and(x, y) → bigint

  Returns the bitwise AND result of x and y in binary complement.

  ```
  select bitwise_and(8, 7); -- 0
  ```

- bitwise_not(x) → bigint

  Returns the result of x bitwise NOT in binary complement.

  ```
  select bitwise_not(8);-- -9
  ```

- bitwise_or(x, y) → bigint

  Returns the bitwise OR result of x and y in binary complement.

  ```
  select bitwise_or(8,7);-- 15
  ```

- bitwise_xor(x, y) → bigint

  Returns the bitwise XOR result of x and y in binary complementary code format.

  ```
  SELECT bitwise_xor(19,25); -- 10
  ```

- bitwise_left_shift(*value*, *shift*) → [same as value]

  Description: Returns the value that is shifted leftwards by **shift** bits.

  ```
  SELECT bitwise_left_shift(1, 2); -- 4
  SELECT bitwise_left_shift(5, 2); -- 20
  SELECT bitwise_left_shift(0, 1); -- 0
  SELECT bitwise_left_shift(20, 0); -- 20
  ```

- bitwise_right_shift(value, shift) → [same as value]

  Description: Returns the value that is shifted rightwards by **shift** bits.

  ```
  SELECT bitwise_right_shift(8, 3); -- 1
  SELECT bitwise_right_shift(9, 1); -- 4
  SELECT bitwise_right_shift(20, 0); -- 20
  SELECT bitwise_right_shift(0, 1); -- 0
  -- If the value is shifted rightwards by more than 64 bits, return 0.
  SELECT bitwise_right_shift( 12, 64); -- 0
  ```

- bitwise_right_shift_arithmetic(*value*, *shift*) → [same as value]

  Description: Returns the value that is arithmetically right shifted. When shift is less than 64 bits, the return value is the same as the value returned by **bitwise_right_shift**. When **shift** reaches or exceeds 64 bits, this function

returns **0** if the value is a positive number, and returns **-1** if the value is a negative number.

```
SELECT bitwise_right_shift_arithmetic( 12, 64); --  0
SELECT bitwise_right_shift_arithmetic(-45, 64); -- -1
```

11.13.6.8 Decimal Functions and Operators

DECIMAL Literal

You can use the DECIMAL'xxxxxxx.yyyyyyy' syntax to define literals of the DECIMAL type.

The literal precision of the DECIMAL type will be equal to the number of bits of the literal (including trailing zeros and leading zeros). The range will be equal to the number of digits in the decimal part (including trailing zeros).

Example Literal	Data Type
DECIMAL '0'	DECIMAL(1)
DECIMAL '12345'	DECIMAL(5)
DECIMAL '0000012345.1234500000'	DECIMAL(20, 10)

Binary Arithmetic Decimal Operator

Standard mathematical operators are supported. The following table describes the rules for calculating the precision and range of the results. Assume that the type of x is DECIMAL(xp, xs) and the type of y is DECIMAL(yp, ys).

Calculation	Result Type Precision	Result Type Range
x + y and x - y	min(38, 1 + min(xs, ys) + min(xp - xs, yp - ys))	max(xs, ys)
x * y	min(38, xp + yp)	xs + ys
x / y	min(38, xp + ys + max(0, ys-xs))	max(xs, ys)
x % y	min(xp - xs, yp - ys) + max(xs, bs)	max(xs, ys)

If the mathematical result of the operation cannot be accurately represented by the precision and range of the result data type, the exception occurs: Value is out of range.

When an operation is performed on a decimal type with different ranges and precisions, the value is first cast to a common supertype. For types close to the maximum representable precision (38), this may cause a "value out of range" error when an operand does not conform to the public supertype. For example, the common supertype of decimal (38, 0) and decimal (38, 1) is decimal (38, 1),

but some values that comply with decimal (38, 0) cannot be represented as decimal (38, 1).

Comparison Operators

All standard comparison operators and BETWEEN operators apply to DECIMAL types.

Unary Decimal Operators

The operator "-" performs a negative operation. The type of the result is the same as that of the parameter.

11.13.6.9 String Functions and Operators

String Operators

|| Indicates the character connection.

```
SELECT 'he'||'llo'; –hello
```

String Functions

These functions assume that the input string contains valid UTF-8 encoded Unicode code points. They do not explicitly check whether UTF-8 data is valid. For invalid UTF-8 data, the function may return an incorrect result. You can use from_utf8 to correct invalid UTF-8 data.

In addition, these functions operate on Unicode code points, not on characters (or font clusters) visible to users. Some languages combine multiple code points into a single user-perceived character (which is the basic unit of the language writing system), but functions treat each code point as a separate unit.

The lower and upper functions do not perform locale-related, context-related, or one-to-many mappings required by certain languages.

- chr(n) → varchar

  Description: Returns the value of a character whose Unicode encoding value is **n**.

  ```
  select chr(100); --d
  ```

- char_length(string) → bigint

  For details, see **length(string)**.

- character_length(string) → bigint

  For details, see **length(string)**.

- codepoint(*string*) → integer

  Description: Returns the Unicode encoding of a single character.

  ```
  select codepoint('d'); --100
  ```

- concat(string1, string2) → varchar

  Description: Concatenates strings.

  ```
  select concat('hello','world'); -- helloworld
  ```


- concat_ws(string0, string1, ..., stringN) → varchar

  Description: Concatenates string1, string2, ..., and stringN into a string using string0 as the separator. If string0 is null, the return value is null. If the parameter following the separator is null, the parameter will be skipped during concatenation.

  ```
  select concat_ws(',','hello','world'); -- hello,world
  select concat_ws(NULL,'def'); --NULL
  select concat_ws(',','hello',NULL,'world'); -- hello,world
  select concat_ws(',','hello','','world'); -- hello,,world
  ```

- concat_ws(string0, array(varchar)) → varchar

  Description: Concatenates elements in an array using string0 as the separator. If string0 is null, the return value is null. Any null value in the array will be skipped.

  ```
  select concat_ws(NULL,ARRAY['abc']);--NULL
  select concat_ws(',',ARRAY['abc',NULL,NULL,'xyz']); -- abc,xyz
  select concat_ws(',',ARRAY['hello','world']); -- hello,world
  ```

- decode(binary bin, string charset) →varchar

  Description: Encodes the first parameter into a string based on the specified character set. The supported character sets include UTF-8, UTF-16BE, UTF-16LE, and UTF-16. If the first parameter is null, null is returned.

  ```
  select decode(X'70 61 6e 64 61','UTF-8');
   _col0
  -------
   panda
  (1 row)

  select decode(X'00 70 00 61 00 6e 00 64 00 61','UTF-16BE');
   _col0
  -------
   panda
  (1 row)
  ```

- encode(string str, string charset) →binary

  Description: Encodes a string based on the specified character set.

  ```
  select encode('panda','UTF-8');
      _col0
  ----------------
   70 61 6e 64 61
  (1 row)
  ```

- find_in_set (string str, string strList) →int

  Description: Returns the position of the first occurrence of the string in the comma-separated strList. If a parameter is null, null is returned.

  ```
  select find_in_set('ab', 'abc,b,ab,c,def'); -- 3
  ```

- format_number(number x, int d) →string

  Description: Formats the number *x* to **#,###,###.##**, reserves *d* decimal places, and returns the result as a string.

  ```
  select format_number(541211.212,2); -- 541,211.21
  ```

- format(format,args...) → varchar

  Description: For details, see **Format**.

- locate(string substr, string str, int pos]) →int

  Description: Returns the position of the first occurrence of the substring after the *pos* position in the string. If the condition is not met, **0** is returned.


```
select locate('aaa','bbaaaaa',6);-- 0
select locate('aaa','bbaaaaa',1);-- 3
select locate('aaa','bbaaaaa',4);-- 4
```

- length(string) → bigint

  Description: Returns the length of the string.

  ```
  select length('hello');-- 5
  ```

- levenshtein_distance(string1, string2) → bigint

  Description: Calculates the Levenshtein distance between string1 and string2, that is, the minimum number of single-character edits (insertions, deletions, or substitutions) required to convert string1 to string2.

  ```
  select levenshtein_distance('helo word','hello,world'); -- 3
  ```

- hamming_distance(*string1*, *string2*) → bigint

  Description: Returns the Hamming distance between character strings 1 and 2, that is, the number of different characters in the corresponding positions. Note that the lengths of the two strings must be the same.

  ```
  select hamming_distance('abcde','edcba');-- 4
  ```

- instr(string,substring) → bigint

  Description: Locates the first occurrence of a substring in a string.

  ```
  select instr('abcde', 'cd');--3
  ```

- levenshtein(*string1*, *string2*) → bigint

  For details, see **levenshtein_distance(*string1*, *string2*)**.

- levenshtein_distance(*string1*, *string2*) → bigint

  Description: Returns the Levenshtein edit distance between string 1 and string 2, that is, the minimum number of single-character edits (insertion, deletion, or replacement) required to change string 1 to string 2.

  ```
  select levenshtein_distance('apple','epplea');-- 2
  ```

- lower(string) → varchar

  Description: Converts characters into lowercase letters.

  ```
  select lower('HELLo!');-- hello!
  ```

- lcase(string A) → varchar

  Description: Same as **lower(string)**.

- ltrim(string) → varchar

  Description: Removes spaces at the beginning of a character string.

  ```
  select ltrim('   hello');-- hello
  ```

- lpad(*string*, *size*, *padstring*) → varchar

  Description: Pads the string to the left to resize it using **padstring**. If *size* is less than the length of the string, the result is truncated to *size* characters. The size cannot be negative, and the padding string must not be empty.

  ```
  select lpad('myk',5,'dodo'); -- domyk
  ```

- luhn_check(string) → boolean

  Description: Tests whether a numeric string is valid based on the Luhn algorithm.

  This checksum function, also known as mod 10, is widely used to validate a variety of identification numbers, such as credit card numbers and ID card numbers.

  ```
  select luhn_check('79927398713'); -- true
  select luhn_check('79927398714'); -- false
  ```


- octet_length(string str) → int

  Description: Returns the number of bytes for saving the string encoded using UTF-8.

  ```
  select octet_length('query');--5
  ```

- parse_url(string urlString, string partToExtract [, string keyToExtract]) →string

  Description: Returns the specified part of a URL. The valid value of the **partToExtract** parameter is **HOST**, **PATH**, **QUERY**, **REF**, **PROTOCOL**, **AUTHORITY**, **FILE**, and **USERINFO**. **keyToExtract** is an optional parameter, which is used to select the value corresponding to the key in **QUERY**.

  ```
  select parse_url('https://www.example.com/index.html','HOST');
    _col0
  ----------
   www.example.com
  (1 row)

  -- Query the value of service in QUERY of the URL.
  select parse_url('https://www.example.com/query/index.html?name=panda','QUERY','name');
   _col0
  -------
   panda
  (1 row)
  ```

- position(*substring IN string*) →bigint

  Description: Returns the position of the first occurrence of a substring in the parent string.

  ```
  select position('ab' in 'sssababa');-- 4
  ```

- quote(String text) → string

  Description: Returns a string enclosed in single quotation marks. Strings containing single quotation marks are not supported.

  ```
  select quote('DONT');-- 'DONT'
  select quote(NULL);-- NULL
  ```

- repeat2(string str, int n) → string

  Description: Returns a string obtained by repeating the **str** string for *n* times.

  ```
  select repeat2('abc',4);
     _col0
  --------------
   abcabcabcabc
  (1 row)
  ```

- replace(string, 'a')→ varchar

  Description: Removes the character **a** from the character string.

  ```
  select replace('hello','e');-- hllo
  ```

- replace(string, 'a', 'b')→ varchar

  Description: Replaces all **a** characters in a string with **b**.

  ```
  select replace('hello','l','m');-- hemmo
  ```

- reverse(string) → varchar

  Description: reverses the string.

  ```
  select reverse('hello');-- olleh
  ```

- rpad(*string*, *size*, *padstring*) → varchar

  Description: Pads the string to the right to resize it using *padstring*. If *size* is less than the length of the string, the result is truncated to *size* characters. The size cannot be negative, and the padding string must not be empty.

  ```
  select rpad('myk',5,'dog'); -- mykdo
  ```


- rtrim(string) → varchar

  Description: Removes spaces at the end of a character string.

  ```
  select rtrim('hello world!   ');-- hello world!
  ```

- space(int n) → varchar

  Description: Returns *n* spaces.

  ```
  select space(4);
   _col0
  -------

  (1 row)

  select length(space(4));
   _col0
  -------
      4
  (1 row)
  ```

- split(string, delimiter) → array

  Description: Splits the string by delimiters into an array.

  ```
  select split('a:b:c:d',':');-- [a, b, c, d]
  ```

- split(string, delimiter, limit) → array

  Description: Splits a string into an array by delimiter. **limit** indicates the number of elements. The last element contains all the characters of the last string. **limit** must be a number.

  ```
  select split('a:b:c:d',':',2);-- [a, b:c:d]
  select split('a:b:c:d',':',4);-- [a, b, c, d]
  ```

- split_part(string, delimiter, index) → varchar

  Description: Splits a string into an array by delimiter and extracts the elements whose index value is index. The index starts from 1. If the index exceeds the array length, **NULL** is returned.

  ```
  select split_part('a:b:c:d',':',2); -- b
  select split_part('a:b:c:d',':',5); -- NULL
  ```

- split_to_map (string, entryDelimiter, keyValueDelimiter) → map<varchar, varchar>

  Description: Splits a string into mapped key-value pairs by **entryDelimiter**, and each key-value pair differentiates keys and values by **keyValueDelimiter**.

  ```
  select split_to_map('li:18,wang:17',',',':');--{wang=17, li=18}
  ```

- split_to_multimap(string, entryDelimiter, keyValueDelimiter) -> map(varchar, array(varchar)

  Description: Splits a string by **entryDelimiter** and **keyValueDelimiter** and returns a map. Each key corresponds to a value of the array type. **entryDelimiter** splits a string into key-value pairs, and **keyValueDelimiter** splits a key-value pair into a key and a value.

  ```
  select split_to_multimap('li:18,wang:17,li:19,wang:18',',',':');--{wang=[17, 18], li=[18, 19]}
  ```

- strpos(string, substring) → bigint

  Description: Returns the position of the first occurrence of **substring** in a string. The value starts from 1. If the value is not found, the value **0** is returned. Example:

  ```
  select strpos('hello world!','l'); --3
  select strpos('hello world!','da'); --0
  ```

- str_to_map() For details, see **split_to_map()**.

- substr(string, start) → varchar

  Description: Truncates a character string from the **start** position.

  ```
  select substr('hello world',3);-- llo world
  ```

- substr(string, start, length) → varchar

  Description: Truncates a character string from the **start** position. The truncated length is **length**.

  Generally, it is used to truncate the timestamp format.

  ```
  Select substr('2019-03-10 10:00:00',1,10); --Truncate to March 10, 2019.
  Select substr('2019-03-10 10:00:00',1,7); --Truncate to March 2019.
  ```

- substring(string, start) → varchar

  For details, see **substr(string, start)**.

- substring_index(string A, string delim, int count) → varchar

  Description: If *count* is a positive number, all content before the *count* delimiter from the left is returned. If *count* is a negative number, all content after the *count* delimiter from the right is returned.

  ```
  select substring_index('one.two.three','.',2);
       _col0
  ------------------
   one.two
  (1 row)

  select substring_index('one.two.three','.',-2);
       _col0
  ------------------
   two.three
  (1 row)

  select substring_index('one.two.three','.',0);
   _col0
  -------
   NULL
  (1 row)
  ```

- soundex(string A) →varchar

  Description: Returns code (**soundex**) consisting of four characters to evaluate the similarity of two strings in pronunciation. The rules are as follows:

  Table 11-78 Character mapping rule

Character	Digit
a, e, h, i, o, u, w, and y	0
b. f, p, and v	1
c. g, j, k, q, s, x, and z	2
d and t	3
l	4
m and n	5
r	6

  - Extracts the first letter of a string as the first value of **soundex**.
  - Replaces the latter letters with digits one by one based on the preceding letter mapping rules. If there are consecutive equal numbers, retain only one number and delete.
  - If the result contains more than four digits, the first four digits are used. If the result contains less than four digits, pad 0s to the end.

```
select soundex('Miller');
 _col0
-------
 M460
(1 row)
```

- translate(string|char|varchar input, string|char|varchar from, string|char|varchar to) →varchar

  Description: Replaces the string specified by the **from** parameter with the string specified by the **to** parameter for an input string. If one of the three parameters is null, **NULL** is returned.

```
select translate('aabbcc','bb','BB');
 _col0
--------
 aaBBcc
(1 row)
```

- trim(string) → varchar

  Description: Removes spaces at the beginning and end of a character string.

```
select trim('  hello world!  ');-- hello world!
```

- btrim(String str1,String str2) → varchar

  Description: Removes all characters contained in **str2** from the beginning and end of **str1**.

```
select btrim('hello','hlo');-- e
```

- upper(string) → varchar

  Description: Converts character strings to uppercase letters.

```
select upper('heLLo');-- HELLO
```

- ucase(string A) → varchar

  Description: Same as **upper(string)**.

- base64decode(STRING str)

  Description: Performs Base64 reverse encoding on the character string.

```
SELECT to_base64(CAST('hello world' as varbinary));-- aGVsbG8gd29ybGQ=
select base64decode('aGVsbG8gd29ybGQ=');-- hello world
```

- jaro_distance(STRING str1, STRING str2)

  Description: Compares the similarity between two character strings.

```
select JARO_DISTANCE('hello', 'hell');-- 0.9333333333333332
```

- FNV_HASH(type v)

  Description: Calculates the hash value of a character string.

```
select FNV_HASH('hello');-- -6615550055289275125
```

- word_stem(word) → varchar

  Description: Returns the stem of an English word.

```
select word_stem('greating');-- great
```

- word_stem(*word*, *lang*) → varchar

  Description: Returns the stem of a word in a specified language.


```
select word_stem('ultramoderne','fr');-- ultramodern
```

- translate(*source*, *from*, *to*) → varchar

  Description: Returns the translated source string by replacing the characters found in the source string with the corresponding characters in the target string. If the **from** string contains duplicate items, only the first one is used. If the source character does not exist in the **from** string, the source character is copied without translation. If the index of the matching character in the **from** string exceeds the length of the **to** string, the source character is omitted from the result string.

```
SELECT translate('abcd', '', ''); -- 'abcd'
SELECT translate('abcd', 'a', 'z'); -- 'zbcd'
SELECT translate('abcda', 'a', 'z'); -- 'zbcdz'
SELECT translate('Palhoça', 'ç','c'); -- 'Palhoca'
SELECT translate('abcd', 'a', ''); -- 'bcd'
SELECT translate('abcd', 'a', 'zy'); -- 'zbcd'
SELECT translate('abcd', 'ac', 'z'); -- 'zbd'
SELECT translate('abcd', 'aac', 'zq'); -- 'zbd'
```

Unicode functions

- normalize(string) → varchar

  Description: Returns a standard string in NFC format.

```
select normalize('e');
 _col0
-------
 e
(1 row)
```

- normalize(string, form) → varchar

  Description: Unicode allows you to write the same character in different bytes. For example, **é** consists of **0xC3** and **0xA9**, and **é** consists of **0x65**, **0xCC**, and **0x81**.

  normalize() returns a standard string based on the Unicode standard formats (including NFC, NFD, NFKC, and NFKD) specified by the parameter format. If no parameter format is specified, NFC is used by default.

```
select to_utf8('é');
 _col0
-------
 c3 a9
(1 row)

select to_utf8('é');
  _col0
----------
 65 cc 81
(1 row)

select normalize('é',NFC)=normalize('é',NFC);
 _col0
-------
 true
(1 row)
```

- to_utf8(string) → varbinary

  Description: Encodes a string into a UTF-8 string.

```
select to_utf8('panda');
     _col0
----------------
 70 61 6e 64 61
(1 row)
```


- from_utf8(binary) → varchar

  Description: Encodes a binary string into a UTF-8 string. An invalid UTF-8 sequence will be replaced by the Unicode character U+FFFD.

```
select from_utf8(X'70 61 6e 64 61');
 _col0
-------
 panda
(1 row)
```

- from_utf8(binary, replace) → varchar

  Description: Encodes a binary string into a UTF-8 string. An invalid UTF-8 sequence will be replaced by the **replace** parameter. The value of the **replace** parameter must be a single character or empty to prevent invalid characters from being removed.

```
select from_utf8(X'70 61 6e 64 61 b1','!');
 _col0
--------
 panda!
(1 row)
```

11.13.6.10 Regular Expressions

Overview

All regular expression functions use Java-style syntax, except in the following cases:

- Use the multi-line mode (through (? m) flag enabling), only **\n** is identified as a line terminator. In addition, it does not support (? d) Flag. Therefore, it cannot be used.
- In case-sensitive mode (through (? i) flag enabling), the unicode mode is always used. In addition, context-sensitive matching and local sensitive matching are not supported. In addition, it does not support (? u) flag.
- The Surrogate Pair encoding mode is not supported. For example, **\ uD800 \ uDC00** is not considered as U + 10000 and must be specified as **\ x {10000}**.
- The boundary character (**\b**) cannot be handled correctly because it is a non-spaced marker without a base character.
- \Q and \E are not supported in character classes (such as [A-Z123]). They are processed as text.
- Unicode characters (\ p {prop}) are supported. The differences are as follows:
  - All underscores in the name must be deleted. For example, use **OldItalic** instead of **Old_Italic**.
  - You must specify a script without the prefix **Is**, **script =**, or **sc =**. Example: **\p {Hiragana}**
  - The **In** prefix must be used to specify a block. The prefix **block =** or **blk =** is not supported. Example: **\p{Mongolian}**
  - You must specify a category without the prefix **Is**, **general_category =**, or **gc =**. Example: **\p{L}**
  - The binary attribute must be specified directly, not **Is**. Example: **\p{NoncharacterCodePoint}**

Function

- regexp_count(*string*, *pattern*) → bigint

  Description: Returns the number of pattern matches in a string.

  ```
  SELECT regexp_count('1a 2b 14m', '\s*[a-z]+\s*'); -- 3
  ```

- regexp_extract_all(string, pattern) -> array(varchar)

  Description: Returns all matched substrings in array format.

  ```
  SELECT regexp_extract_all('1a 2b 14m','\d+');-- [1, 2, 14]
  ```

- regexp_extract_all(string, pattern, group) -> array(varchar)

  Description: When the pattern contains multiple groups, group is used to return all substrings that meet the **captured group** conditions.

  ```
  SELECT regexp_extract_all('1a 2b 14m','(\d+)([a-z]+)',2);-- [a, b, m]
  ```

- regexp_extract(string, pattern) → varchar

  Description: Returns the first substring that matches the regular expression pattern in a string.

  ```
  SELECT regexp_extract('1a 2b 14m','\d+');-- 1
  ```

- regexp_extract(string, pattern, group) → varchar

  Description: When the pattern contains multiple groups, **group** is used to specify the first substring that meets **captured group**.

  ```
  SELECT regexp_extract('1a 2b 14m','(\d+)([a-z]+)',2);-- 'a'
  ```

- regexp_like(string, pattern) → boolean

  Description: Checks whether a string contains substrings that meet the regular expression. If yes, **true** is returned.

  ```
  SELECT regexp_like('1a 2b 14m','\d+b');-- true
  ```

- regexp_position(string, pattern) → integer

  Description: Returns the index that matches the pattern for the first time in a string. If no index is matched, returns **-1**.

  ```
  SELECT regexp_position('I have 23 apples, 5 pears and 13 oranges', '\b\d+\b'); -- 8
  ```

- regexp_position(string, pattern, start) → integer

  Description: Returns the index of the item that matches the pattern for the first time starting from the **start** index (included). If no index is matched, returns **-1**.

  ```
  SELECT regexp_position('I have 23 apples, 5 pears and 13 oranges', '\b\d+\b', 5); -- 8
  SELECT regexp_position('I have 23 apples, 5 pears and 13 oranges', '\b\d+\b', 12); -- 19
  ```

- regexp_position(string, pattern, start, occurrence) → integer

  Description: Returns the index of the item that matches the pattern for the **occurrence** time starting from the **start** index (included). If no index is matched, returns **-1**.

  ```
  SELECT regexp_position('I have 23 apples, 5 pears and 13 oranges','\b\d+\b',12,1);-- 19
  SELECT regexp_position('I have 23 apples, 5 pears and 13 oranges','\b\d+\b',12,2);-- 31
  SELECT regexp_position('I have 23 apples, 5 pears and 13 oranges','\b\d+\b',12,3);-- -1
  ```

- regexp_replace(string, pattern) → varchar

  Description: Removes substrings that meet the regular expression from the target string.

  ```
  SELECT regexp_replace('1a 2b 14m','\d+[ab] ');-- '14m'
  ```

- regexp_replace(string, pattern, replacement) → varchar

Description: Replaces the substring that meets the regular expression in the target string with *replacement*. If the *replacement* contains the character **$**, use **\$** to escape the character. During replacement, you can use **$g** to reference a capture group for a numbered group and **${name}** to reference a capture group for a named group.

```
SELECT regexp_replace('1a 2b 14m','(\d+)([ab]) ','3c$2 ');-- '3ca 3cb 14m'
```

- regexp_replace(string, pattern, function) → varchar

  Description: Replaces each instance of the substring that matches the regular expression pattern in the string with function. For each match, the **captured group** passed as an array calls the **lambda** expression function. The capture group ID starts from 1. The entire match is not grouped (brackets enclose the entire expression if necessary).

  ```
  SELECT regexp_replace('new york','(\w)(\w*)',x->upper(x[1])||lower(x[2]));--'New York'
  ```

- regexp_split(string, pattern) -> array(varchar)

  Description: Splits a string using the regular expression pattern and returns an array. The following empty character string is reserved:

  ```
  SELECT regexp_split('1a 2b 14m','\s*[a-z]+\s*');-- [1, 2, 14, ]
  ```

11.13.6.11 Binary Functions and Operators

Binary Operators

|| The operator performs the join.

Binary Functions

- length(binary) → bigint

  Return the byte length of **binary**.

  ```
  select length(x'00141f');-- 3
  ```

- concat(binary1, ..., binaryN) → varbinary

  Concatenates *binary1*, *binary2*, and *binaryN*. This function returns the same function as the SQL standard connector ||.

  ```
  select concat(X'32335F',x'00141f'); -- 32 33 5f 00 14 1f
  ```

- to_base64(binary) → varchar

  Encodes *binary* to a Base64 character string.

  ```
  select to_base64(CAST('hello world' as binary)); -- aGVsbG8gd29ybGQ=
  ```

- from_base64(string) → varbinary

  Decode the Base64-encoded string as varbinary.

  ```
  select from_base64('helloworld'); --  85 e9 65 a3 0a 2b 95
  ```

- unbase64(string) → varbinary

  Decode the Base64-encoded string as varbinary.

  ```
  SELECT from_base64('helloworld'); --  85 e9 65 a3 0a 2b 95
  ```

- to_base64url(binary) → varchar

  Use URL security characters to encode **binary** to a base64 character string.

  ```
  select to_base64url(x'555555');  -- VVVV
  ```

- from_base64url(string) → varbinary

Use the URL security character to decode the Base64-encoded **string** into binary data.

```
select from_base64url('helloworld'); --  85 e9 65 a3 0a 2b 95
```

- to_hex(binary) → varchar

  Encode the **binary** to a hexadecimal string.

  ```
  select to_hex(x'15245F');  --  15245F
  ```

- from_hex(string) → varbinary

  Decodes a hexadecimal string into binary data.

  ```
  select from_hex('FFFF'); --  ff ff
  ```

- to_big_endian_64(*bigint*) → varbinary

  Encodes a number of the bigint type into a 64-bit big-endian complement.

  ```
  select to_big_endian_64(1234);
           _col0
  -------------------------
   00 00 00 00 00 00 04 d2
  (1 row)
  ```

- from_big_endian_64(*binary*) → bigint

  The binary code in 64-bit big-endian complement format is decoded as a number of the bigint type.

  ```
  select from_big_endian_64(x'00 00 00 00 00 00 04 d2');
   _col0
  -------
    1234
  (1 row)
  ```

- to_big_endian_32(*integer*) → varbinary

  Encodes a number of the bigint type into a 32-bit big-endian complement.

  ```
  select to_big_endian_32(1999);
     _col0
  -------------
   00 00 07 cf
  (1 row)
  ```

- from_big_endian_32(*binary*) → integer

  The 32-bit big-endian two's complement format is decoded into a number of the bigint type.

  ```
  select from_big_endian_32(x'00 00 07 cf');
   _col0
  -------
    1999
  (1 row)
  ```

- to_ieee754_32(*real*) → varbinary

  According to the IEEE 754 algorithm, a single-precision floating-point number is encoded into a 32-bit big-endian binary block.

  ```
  select to_ieee754_32(3.14);
     _col0
  -------------
   40 48 f5 c3
  (1 row)
  ```

- from_ieee754_32(*binary*) → real

  Decodes the 32-bit big-endian binary in IEEE 754 single-precision floating-point format.

  ```
  select from_ieee754_32(x'40 48 f5 c3');
   _col0
  -------
  ```



```
 3.14
(1 row)
```

- to_ieee754_64(*double*) → varbinary

  According to the IEEE 754 algorithm, a double-precision floating point number is encoded into a 64-bit big-endian binary block.

```
select to_ieee754_64(3.14);
 _col0
-------------------------
 40 09 1e b8 51 eb 85 1f
(1 row)
```

- from_ieee754_64(*binary*) → double

  Decodes 64-bit big-endian binary in IEEE 754 single-precision floating-point format.

```
 select from_ieee754_64(X'40 09 1e b8 51 eb 85 1f');
 _col0
-------
  3.14
(1 row)
```

- lpad(*binary*, *size*, *padbinary*) → varbinary

  Left-padded binary to adjust byte size using padbinary. If *size* is less than the length of the binary file, the result will be truncated to *size* characters. The value of *size* cannot be negative, and the value of *padbinary* cannot be empty.

```
select lpad(x'15245F', 11,x'15487F') ; -- 15 48 7f 15 48 7f 15 48 15 24 5f
```

- rpad(*binary*, *size*, *padbinary*) → varbinary

  Right-padded binary to use padbinary to resize bytes. If *size* is less than the length of the binary file, the result will be truncated to *size* characters. The value of *size* cannot be negative, and the value of *padbinary* cannot be empty.

```
SELECT rpad(x'15245F', 11,x'15487F'); -- 15 24 5f 15 48 7f 15 48 7f 15 48
```

- crc32(*binary*) → bigint

  Calculate the CRC 32 value of the binary block.

- md5(*binary*) → varbinary

  Calculates the MD 5 hash value of a binary block.

- sha1(*binary*) → varbinary

  Calculates the SHA 1 hash value of a binary block.

- sha2(*string, integer*) → string

  SHA2 is a standard cryptographic hash algorithm used to convert a variable-length string into a string. Its output can be 224 bits, 256 bits, 384 bits, or 512 bits, corresponding to SHA-224, SHA-256, SHA-384 and SHA512, respectively.

- sha256(*binary*) → varbinary

  Calculates the SHA 256 hash value of a binary block.

- sha512(*binary*) → varbinary

  Calculates the SHA 512 hash value of a binary block.

- xxhash64(*binary*) → varbinary

  Calculates the XXHASH 64 hash value of a binary block.

- spooky_hash_v2_32(*binary*) → varbinary

  Calculates the 32-bit SpookyHashV2 hash value of a binary block.

- spooky_hash_v2_64(*binary*) → varbinary

Calculates the 64-bit SpookyHashV2 hash value of a binary block.

- hmac_md5(*binary*, *key*) → varbinary

  Use the given key to calculate the HMAC value of the binary block (using MD5).

- hmac_sha1(*binary*, *key*) → varbinary

  Use the given key to calculate the HMAC value of the binary block (using SHA1).

- hmac_sha256(*binary*, *key*) → varbinary

  Use the given key to calculate the HMAC value of the binary block (using SHA256).

- hmac_sha512(*binary*, *key*) → varbinary

  Use the given key to calculate the HMAC value of the binary block (using SHA512).

NOTICE

The CRC32, MD5, and SHA1 algorithms have been found to have vulnerabilities. Do not use them for cryptography.

11.13.6.12 JSON Functions and Operators

- Cast to JSON

```
SELECT CAST(9223372036854775807 AS JSON); -- JSON '9223372036854775807'
```

- Cast from JSON

```
SELECT CAST(JSON '[1,23,456]' AS ARRAY(INTEGER)); -- [1, 23, 456]
```

JSON Function

NOTE

The conversion from NULL to JSON cannot be simply implemented. Converting from a separate NULL produces a SQLNULL instead of JSON'null'. However, when converted from an array or Map containing NULL, the generated JSON will contain NULL.

When converted from ROW to JSON, the result is a JSON array, not a JSON object. This is because for rows in SQL, the location is more important than the name.

The value can be converted from BOOLEAN, TINYINT, SMALLINT, INTEGER, BIGINT, REAL, DOUBLE, or VARCHAR. If the element type of an array is one of the supported types, the key type of a map is VARCHAR, and the value type of a map is one of the supported types, or the type of each field in a row is one of the supported types, the array can be converted from ARRAY, MAP, or ROW. The following example shows the behavior of the transformation:

```
SELECT CAST(NULL AS JSON);-- NULL
SELECT CAST(1 AS JSON);-- JSON '1'
SELECT CAST(9223372036854775807 AS JSON);-- JSON '9223372036854775807'
SELECT CAST('abc' AS JSON);-- JSON '"abc"'
SELECT CAST(true AS JSON);-- JSON 'true'
SELECT CAST(1.234 AS JSON);-- JSON '1.234'
SELECT CAST(ARRAY[1, 23, 456] AS JSON);-- JSON '[1,23,456]'
SELECT CAST(ARRAY[1, NULL, 456] AS JSON);-- JSON '[1,null,456]'
SELECT CAST(ARRAY[ARRAY[1, 23], ARRAY[456]] AS JSON);-- JSON '[[1,23],[456]]'
SELECT CAST(MAP(ARRAY['k1', 'k2', 'k3'], ARRAY[1, 23, 456]) AS JSON);-- JSON '{"k1":1,"k2":23,"k3":456}'
```



```
SELECT CAST(CAST(ROW(123, 'abc', true) AS ROW(v1 BIGINT, v2 VARCHAR, v3 BOOLEAN)) AS JSON);--
JSON '[123,"abc",true]'
```

JSON-to-other Types

```
SELECT CAST(JSON 'null' AS VARCHAR);-- NULL
SELECT CAST(JSON '1' AS INTEGER);-- 1
SELECT CAST(JSON '9223372036854775807' AS BIGINT);-- 9223372036854775807
SELECT CAST(JSON '"abc"' AS VARCHAR);-- abc
SELECT CAST(JSON 'true' AS BOOLEAN);-- true
SELECT CAST(JSON '1.234' AS DOUBLE);-- 1.234
SELECT CAST(JSON '[1,23,456]' AS ARRAY(INTEGER));-- [1, 23, 456]
SELECT CAST(JSON '[1,null,456]' AS ARRAY(INTEGER));-- [1, NULL, 456]
SELECT CAST(JSON '[[1,23],[456]]' AS ARRAY(ARRAY(INTEGER)));-- [[1, 23], [456]]
SELECT CAST(JSON '{"k1":1, "k2":23, "k3":456}' AS MAP(VARCHAR, INTEGER));-- {k1=1, k2=23, k3=456}
SELECT CAST(JSON '{"v1":123, "v2":"abc","v3":true}' AS ROW(v1 BIGINT, v2 VARCHAR, v3 BOOLEAN));--
{v1=123, v2=abc, v3=true}
SELECT CAST(JSON '[123, "abc",true]' AS ROW(v1 BIGINT, v2 VARCHAR, v3 BOOLEAN));-- {value1=123,
value2=abc, value3=true}
SELECT CAST(JSON'[[1, 23], 456]'AS ARRAY(JSON));-- [JSON '[1,23]', JSON '456']
SELECT CAST(JSON'{"k1": [1, 23], "k2": 456}'AS MAP(VARCHAR,JSON));-- {k1 = JSON '[1,23]', k2 = JSON
'456'}
SELECT CAST(JSON'[null]'AS ARRAY(JSON));-- [JSON 'null']
```

NOTE

JSON arrays and JSON objects are supported during conversion from JSON to ROW.

JSON arrays can have mixed element types, and JSON Maps can have mixed value types. This makes it impossible to convert it to SQL arrays and maps in some cases. To solve this problem, HetuEngine supports partial conversion of arrays and maps.

```
SELECT CAST(JSON'[[1, 23], 456]'AS ARRAY(JSON));-- [JSON '[1,23]', JSON '456']
SELECT CAST(JSON'{"k1": [1, 23], "k2": 456}'AS MAP(VARCHAR,JSON));-- {k1 = JSON '[1,23]', k2 =
JSON '456'}
SELECT CAST(JSON'[null]'AS ARRAY(JSON));-- [JSON 'null']
```

- is_json_scalar(json) → boolean

  Check whether JSON is a scalar (that is, JSON number, JSON character string, true, false, or null).

  ```
  select is_json_scalar(json'[1,22]'); -- false
  ```

- json_array_contains(json, value) → boolean

  Checks whether a value is contained in **json**.

  ```
  select json_array_contains(json '[1,23,44]',23); -- true
  ```

- json_array_get(json_array, index) → json

  NOTICE

  The semantics of the function has been broken. If the extracted element is a string, it will be converted to an invalid JSON value that is not correctly enclosed in quotation marks (the value will not be enclosed in quotation marks, and any internal quotation marks will not be escaped). You are not advised to use this function. The function cannot be corrected without affecting existing usage and may be deleted in future versions.

  Returns the JSON element at the specified index position. The index starts from 0.

  ```
  SELECT json_array_get('["a", [3, 9], "c"]', 0); -- JSON 'a' (invalid JSON)
  SELECT json_array_get('["a", [3, 9], "c"]', 1); -- JSON '[3,9]'
  ```


The index page supports negative numbers, indicating that the index page starts from the last element. The value **-1** indicates the last element. If the index length exceeds the actual length, **null** is returned.

```
SELECT json_array_get('["c", [3, 9], "a"]', -1); -- JSON 'a' (invalid JSON)
SELECT json_array_get('["c", [3, 9], "a"]', -2); -- JSON '[3,9]'
```

If the JSON element at the specified index does not exist, NULL is returned.

```
SELECT json_array_get('[]', 0);                 -- NULL
SELECT json_array_get('["a", "b", "c"]', 10);  -- NULL
SELECT json_array_get('["c", "b", "a"]', -10); -- NULL
```

- json_array_length(json) → bigint

  Returns the length of **json**.

  ```
  SELECT json_array_length(json '[1,2,3,4]'); -- 4
  SELECT json_array_length('[1, 2, 3]'); -- 3
  ```

- get_json_object(string json,string json_path);

  Captures information in **json** based on the **json_path** format.

  ```
  SELECT get_json_object('{"id": 1, "value":"xxx"}', '$.value');  -- "xxx"
  ```

- json_extract(json, json_path) → json

  Captures information in **json** based on the **json_path** format.

  ```
  SELECT json_extract(json '{"id": 1, "value":"xxx"}', '$.value');-- JSON "xxx"
  ```

- json_extract_scalar(json, json_path) → varchar

  The function is the same as that of **json_extract**. The return value is varchar.

  ```
  SELECT json_extract_scalar(json '{"id": 1, "value": "xxx"}', '$.value'); -- xxx
  ```

- json_format(json) → varchar

  Converts a JSON value to a serialized JSON text. This is the inverse function of json_parse.

  ```
  SELECT JSON_format(json '{"id": 1, "value":"xxx"}'); -- {"id":1, "value":"xxx"}
  ```

  Notes:

  json_format and CAST(json AS VARCHAR) have completely different semantics.

  json_format serializes the input JSON value into JSON text that complies with the 7159 standard. The JSON value can be a JSON object, JSON array, JSON string, JSON number, true, false, or null:

  ```
  SELECT json_format(JSON '{"a": 1, "b": 2}'); -- '{"a":1,"b":2}'
  SELECT json_format(JSON '[1, 2, 3]'); -- '[1,2,3]'
  SELECT json_format(JSON '"abc"'); -- '"abc"'
  SELECT json_format(JSON '42'); -- '42'
  SELECT json_format(JSON 'true'); -- 'true'
  SELECT json_format(JSON 'null'); -- 'null'
  ```

  CAST(json AS VARCHAR) converts the JSON value to the corresponding SQL VARCHAR value. For JSON strings, JSON numbers, true, false, or null, the conversion behavior is the same as that of the corresponding SQL type. JSON objects and JSON arrays cannot be converted to VARCHAR.

  ```
  SELECT CAST(JSON '{"a": 1, "b": 2}' AS VARCHAR); -- NULL
  SELECT CAST(JSON '[1, 2, 3]' AS VARCHAR); -- NULL
  SELECT CAST(JSON '"abc"' AS VARCHAR); -- 'abc'; Note the double quote is gone
  SELECT CAST(JSON '42' AS VARCHAR); -- '42'
  SELECT CAST(JSON 'true' AS VARCHAR); -- 'true'
  SELECT CAST(JSON 'null' AS VARCHAR); -- NULL
  ```

- json_parse(string) → json

  Contrary to **json_format(json)**, convert a JSON character string to JSON.

json_parse and **json_extract** are used together to parse JSON character strings in data tables.

```
select JSON_parse('{"id": 1, "value":"xxx"}'); -- json {"id":1, "value":"xxx"}
```

- json_size(json, json_path) → bigint

  It is similar to **json_extract**, but the number of objects in JSON is returned.

```
SELECT json_size('{ "x": {"a": 1, "b": 2} }', '$.x'); => 2
SELECT json_size('{ "x": [1, 2, 3] }', '$.x'); =>3
SELECT json_size('{ "x": {"a": 1, "b": 2} }', '$.x.a'); => 0
```

11.13.6.13 Date and Time Functions and Operators

Date and Time Operators

Operator	Example	Result
+	date '2012-08-08' + interval '2' day	2012-08-10
+	time '01:00' + interval '3' hour	04:00:00.000
+	timestamp '2012-08-08 01:00' + interval '29' hour	2012-08-09 06:00:00.000
+	timestamp '2012-10-31 01:00' + interval '1' month	2012-11-30 01:00:00.000
+	interval '2' day + interval '3' hour	2 03:00:00.000
+	interval '3' year + interval '5' month	3-5
-	date '2012-08-08' - interval '2' day	2012-08-06
-	time '01:00' - interval '3' hour	22:00:00.000
-	timestamp '2012-08-08 01:00' - interval '29' hour	2012-08-06 20:00:00.000
-	timestamp '2012-10-31 01:00' - interval '1' month	2012-09-30 01:00:00.000
-	interval '2' day - interval '3' hour	1 21:00:00.000
-	interval '3' year - interval '5' month	2-7

Time Zone Conversion

Operator: **AT TIME ZONE** sets the time zone of a timestamp.

```
SELECT timestamp '2012-10-31 01:00 UTC';-- 2012-10-31 01:00:00.000 UTC
SELECT timestamp '2012-10-31 01:00 UTC' AT TIME ZONE 'Asia/Singapore'; -- 2012-10-30 09:00:00.000
Asia/Singapore
```

Date/Time Functions

- current_date -> date

  Returns the current date (UTC time zone).


```
select current_date; -- 2020-07-25
```

- current_time -> time with time zone

  Returns the current time (UTC time zone).

  ```
  select current_time;-- 16:58:48.601+08:00
  ```

- current_timestamp -> timestamp with time zone

  Returns the current timestamp (current time zone).

  ```
  select current_timestamp; -- 2020-07-25 11:50:27.350 Asia/Singapore
  ```

- current_timezone() → varchar

  Returns the current time zone.

  ```
  select current_timezone();-- Asia/Singapore
  ```

- date(x) → date

  Converts a date literal to a variable of the date type.

  ```
  select date('2020-07-25');-- 2020-07-25
  ```

- from_iso8601_timestamp(string) → timestamp with time zone

  Converts a timestamp literal in ISO 8601 format into a timestamp variable with a time zone.

  ```
  SELECT from_iso8601_timestamp('2020-05-11');-- 2020-05-11 00:00:00.000 Asia/Singapore
  SELECT from_iso8601_timestamp('2020-05-11T11:15:05'); -- 2020-05-11 11:15:05.000 Asia/Singapore
  SELECT from_iso8601_timestamp('2020-05-11T11:15:05.055+01:00');-- 2020-05-11 11:15:05.055 +01:00
  ```

- from_iso8601_date(string) → date

  Converts a date literal in ISO 8601 format into a variable of the date type.

  ```
  SELECT from_iso8601_date('2020-05-11');-- 2020-05-11
  SELECT from_iso8601_date('2020-W10');-- 2020-03-02
  SELECT from_iso8601_date('2020-123');-- 2020-05-02
  ```

- from_unixtime(unixtime) → timestamp with time zone

  Converts a UNIX timestamp to a timestamp variable (current time zone).

  ```
  Select FROM_UNIXTIME(1.595658735E9); -- 2020-07-25 14:32:15.000 Asia/Singapore
  Select FROM_UNIXTIME(875996580); --1997-10-05 04:23:00.000 Asia/Singapore
  ```

- from_unixtime(unixtime, string) → timestamp with time zone

  Converts a UNIX timestamp into a timestamp variable. The time zone option can be contained.

  ```
  select from_unixtime(1.595658735E9, 'Asia/Singapore');-- 2020-07-25 14:32:15.000 Asia/Singapore
  ```

- from_unixtime(unixtime, hours, minutes) → timestamp with time zone

  Converts a UNIX timestamp to a timestamp variable with a time zone. **hours** and **minutes** indicate the time zone offsets.

  ```
  select from_unixtime(1.595658735E9, 8, 30);-- 2020-07-25 14:32:15.000 +08:30
  ```

- localtime -> time

  Obtains the current time

  ```
  select localtime;-- 14:16:13.096
  ```

- localtimestamp -> timestamp

  Obtains the current timestamps.

  ```
  select localtimestamp;-- 2020-07-25 14:17:00.567
  ```

- months_between(date1, date2) -> double

  Return the number of months between *date1* and *date2*. If *date1* is later than *date2*, the result is a positive number. Otherwise, the result is a negative number. If the days of the two dates are the same, the result is an integer.

Otherwise, the decimal part is calculated based on the difference between the hour, minute, and second (31 days of each month). The type of *date1* and *date2* can be date, timestamp, or a string in the yyyy-MM-dd or yyyy-MM-dd HH:mm:ss format.

```
select months_between('2020-02-28 10:30:00', '2021-10-30');-- -20.05040323
select months_between('2021-01-30', '2020-10-30'); -- 3.0
```

- now() → timestamp with time zone

  Obtains the current time, which is the alias of **current_timestamp**.

  ```
  select now();-- 2020-07-25 14:39:39.842 Asia/Singapore
  ```

- unix_timestamp()

  Obtains the current **unix** timestamp.

  ```
  select unix_timestamp(); -- 1600930503
  ```

- to_iso8601(x) → varchar

  Converts *x* into a character string in the ISO 8601 format. x can be DATE or TIMESTAMP [with time zone].

  ```
  select to_iso8601(date '2020-07-25'); -- 2020-07-25
  select to_iso8601(timestamp '2020-07-25 15:22:15.214'); -- 2020-07-25T15:22:15.214
  ```

- to_milliseconds(interval) → bigint

  Obtains the number of milliseconds since 00:00 on the current day.

  ```
  select to_milliseconds(interval '8' day to second);-- 691200000
  ```

- to_unixtime(timestamp) → double

  Converts the timestamp to the UNIX time.

  ```
  select to_unixtime(cast('2020-07-25 14:32:15.147' as timestamp));-- 1.595658735147E9
  ```

- trunc(string date, string format) →string

  Truncates a date value based on the format. The supported format is MONTH/MON/MM or YEAR/YYYY/YY, QUARTER/Q

  ```
  select trunc(date '2020-07-08','yy');-- 2020-01-01
  select trunc(date '2020-07-08','MM');-- 2020-07-01
  ```

  NOTE

  You can use the parentheses () when using the following SQL standard functions:

  - current_date
  - current_time
  - current_timestamp
  - localtime
  - Localtimestamp

  For example: **select current_date ();**

Truncation Function

Similar to the operation of reserving decimal places, the **date_trunc** function supports the following units:

Unit	Value After Truncation
second	2001-08-22 03:04:05.000
minute	2001-08-22 03:04:00.000

Unit	Value After Truncation
hour	2001-08-22 03:00:00.000
day	2001-08-22 00:00:00.000
week	2001-08-20 00:00:00.000
month	2001-08-01 00:00:00.000
quarter	2001-07-01 00:00:00.000
year	2001-01-01 00:00:00.000

In the preceding example, the timestamp **2001-08-22 03:04:05.321** is used as the input.

date_trunc(unit, x) → [same as input]

Returns the value of *x* after the **unit**.

```
select date_trunc('hour', timestamp '2001-08-22 03:04:05.321'); -- 2001-08-22 03:00:00.000
```

Interval Functions

The functions in this chapter support the following interval units:

Unit	Description
second	Seconds
minute	Minutes
hour	Hours
day	Days
week	Weeks
month	Months
quarter	Quarters of a year
year	Years

- date_add(unit, value, timestamp) → [same as input]

  Add *value* units to timestamp. If you want to perform the subtraction operation, you can assign a negative value to the *value*.

  ```
  SELECT date_add('second', 86, TIMESTAMP '2020-03-01 00:00:00');-- 2020-03-01 00:01:26
  SELECT date_add('hour', 9, TIMESTAMP '2020-03-01 00:00:00');-- 2020-03-01 09:00:00
  SELECT date_add('day', -1, TIMESTAMP '2020-03-01 00:00:00 UTC');-- 2020-02-29 00:00:00 UTC
  ```

- date_diff(unit, timestamp1, timestamp2) → bigint

  Returns the value of timestamp2 minus timestamp1. The unit of the value is *unit*.

The value of *unit* is a character string. For example, **day**, **week**, and **year**.

```
SELECT date_diff('second', TIMESTAMP '2020-03-01 00:00:00', TIMESTAMP '2020-03-02 00:00:00');-- 86400
SELECT date_diff('hour', TIMESTAMP '2020-03-01 00:00:00 UTC', TIMESTAMP '2020-03-02 00:00:00 UTC');-- 24
SELECT date_diff('day', DATE '2020-03-01', DATE '2020-03-02');-- 1
SELECT date_diff('second', TIMESTAMP '2020-06-01 12:30:45.000', TIMESTAMP '2020-06-02 12:30:45.123');-- 86400
SELECT date_diff('millisecond', TIMESTAMP '2020-06-01 12:30:45.000', TIMESTAMP '2020-06-02 12:30:45.123');-- 86400123
```

- adddate(date, bigint) → [same as input]

  Description: Date addition. The input type can be date or timestamp, indicating that the date is added or deducted. If the input type is subtraction, the value of bigint is negative.

  ```
  select ADDDATE(timestamp '2020-07-04 15:22:15.124',-5);-- 2020-06-29 15:22:15.124
  select ADDDATE(date '2020-07-24',5); -- 2020-07-29
  ```

Duration Function

The duration can use the following units:

Unit	Description
ns	nanosecond
us	microsecond
ms	millisecond
s	second
m	minute
h	hour
d	day

parse_duration(string) → interval

```
SELECT parse_duration('42.8ms'); -- 0 00:00:00.043
SELECT parse_duration('3.81 d'); -- 3 19:26:24.000
SELECT parse_duration('5m'); -- 0 00:05:00.000
```

MySQL Date Functions

The formatted strings that are compatible with the MySQL **date_parse** and **str_to_date** methods in this section.

- date_format(timestamp, format) → varchar

  Uses **format** to format **timestamp**.

  ```
  select date_format(timestamp '2020-07-22 15:00:15', '%Y/%m/%d');-- 2020/07/22
  ```

- date_parse(string, format) → timestamp

  Parses the date literals using **format**.

  ```
  select date_parse('2020/07/20', '%Y/%m/%d');-- 2020-07-20 00:00:00.000
  ```


The following table is based on the MySQL manual and describes various format descriptors.

Format Descriptor	Description
%a	Day in a week (Sun .. Sat)
%b	Month (Jan .. Dec)
%c	Month (1 .. 12)
%D	Day of the month (0th, 1st, 2nd, 3rd, ...)
%d	Day in the month (01.. 31) (Two digits. Zeros are added before the single digits.)
%e	Day in the month (1 .. 31)
%f	Seconds after the decimal point (6 digits for printing: 000000 .. 999000; 1 - 9 digits for parsing: 0 .. 999999999)
%H	Hour (00 .. 23)
%h	Hour (01.. 12)
%I	Hour (01.. 12)
%i	Minute, number (00 .. (59)
%j	Day of the year (001 .. 366)
%k	Hour (0 .. 23)
%l	Hour (1.. 12)
%M	Month name (January .. December)
%m	Month, number (01 .. 12)
%p	AM or PM
%r	Time in the 12-hour format (hh:mm:ss followed by AM or PM)
%S	Second (00 .. 59)
%s	Second (00 .. 59)
%T	Time in the 24-hour format (hh:mm:ss)
%U	Week (00 .. 53) Sunday is the first day of a week.
%u	Week (00 .. 53) Monday is the first day of a week.
%V	Week (01.. 53) Sunday is the first day of a week. Used together with %X.
%v	Week (01 .. 53) Monday is the first day of a week. Used together with %X.

Format Descriptor	Description
%W	Day of the week (Sunday .. Saturday)
%w	Day of the week (0.. 6) Sunday is the first day of a week.
%X	Year, a four-digit number. The first day is Sunday.
%x	Year, a four-digit number. The first day is Monday.
%Y	Year, a four-digit number.
%y	Year. The value is a two-digit number ranging from 1970 to 2069.
%%	Indicates the character '%'.

Example:

```
select date_format(timestamp '2020-07-25 15:04:00.124','%j day of a year with English suffix (0th, 1st, 2nd, 3rd...),%m month %d day,%p %T %W');
                      _col0
-----------------------------------------------------
 207th day of the year, 25th day of July, PM 15:04:00 Saturday
(1 row)
```

NOTE

These format descriptors are not supported: %D, %U, %u, %V, %w, %X.

- date_format(timestamp, format) → varchar

  Uses **format** to format **timestamp**.
- date_parse(string, format) → timestamp

  Parses the timestamp character string.

  ```
  select date_parse('2020/07/20', '%Y/%m/%d');-- 2020-07-20 00:00:00.000
  ```

Java Date Functions

The formatting strings used in this section are compatible with the Java **SimpleDateFormat** style.

- format_datetime(timestamp, format) → varchar

  Uses **format** to format **timestamp**.
- parse_datetime(string, format) → timestamp with time zone

  Format a string to timestamp with time zone in a specified format.

  ```
  select parse_datetime('1960/01/22 03:04', 'yyyy/MM/dd HH:mm');
                  _col0
  ---------------------------------------
   1960-01-22 03:04:00.000 Asia/Shanghai
  (1 row)
  ```


Common Extraction Functions

Domain	Description
YEAR	year()
QUARTER	quarter()
MONTH	month()
WEEK	week()
DAY	day()
DAY_OF_MONTH	day_of_month()
DAY_OF_WEEK	day_of_week()
DOW	day_of_week()
DAY_OF_YEAR	day_of_year()
DOY	day_of_year()
YEAR_OF_WEEK	year_of_week()
YOW	year_of_week()
HOUR	hour()
MINUTE	minute()
SECOND	second()
TIMEZONE_HOUR	timezone_hour()
TIMEZONE_MINUTE	timezone_minute()

Example:

```
select second(timestamp '2020-02-12 15:32:33.215');-- 33
select timezone_hour(timestamp '2020-02-12 15:32:33.215');-- 8
```

- MONTHNAME(date)

  Description: Obtains the month name.

  ```
  SELECT monthname(timestamp '2019-09-09 12:12:12.000');-- SEPTEMBER
  SELECT monthname(date '2019-07-09');--JULY
  ```

- extract(field FROM x) → bigint

  Description: Returns the field from x. For details about the corresponding field, see the table in this document.

  ```
  select extract(YOW FROM timestamp '2020-02-12 15:32:33.215');-- 2020
  select extract(SECOND FROM timestamp '2020-02-12 15:32:33.215');-- 33
  select extract(DOY FROM timestamp '2020-02-12 15:32:33.215');--43
  ```


Function	Example	Description
SECONDS_ADD(TIMESTAMP date, INT seconds)	SELECT seconds_add(timestamp '2019-09-09 12:12:12.000', 10);	The time is added in seconds.
SECONDS_SUB(TIMESTAMP date, INT seconds)	SELECT seconds_sub(timestamp '2019-09-09 12:12:12.000', 10);	Subtracts time in seconds.
MINUTES_ADD(TIMESTAMP date, INT minutes)	SELECT MINUTES_ADD(timestamp '2019-09-09 12:12:12.000', 10);	Add the time in the unit of minute.
MINUTES_SUB(TIMESTAMP date, INT minutes)	SELECT MINUTES_SUB(timestamp '2019-09-09 12:12:12.000', 10);	The time is subtracted in the unit of minute.
HOURS_ADD(TIMESTAMP date, INT hours)	SELECT HOURS_ADD(timestamp '2019-09-09 12:12:12.000', 1);	Add the time in the unit of hour.
HOURS_SUB(TIMESTAMP date, INT hours)	SELECT HOURS_SUB(timestamp '2019-09-09 12:12:12.000', 1);	The time is subtracted in hours.

- last_day(timestamp) -> date

  Description: Returns the last day of each month based on the specified timestamp.

  ```
  SELECT last_day(timestamp '2019-09-09 12:12:12.000');--  2019-09-30
  SELECT last_day(date '2019-07-09');--2019-07-31
  ```

- add_months(timestamp) -> [same as input]

  Description: Returns the correct date by adding the specified date to the specified month.

  ```
  SELECT add_months(timestamp'2019-09-09 00:00:00.000', 11);-- 2020-08-09 00:00:00.000
  ```

- next_day() (timestamp, string) -> date

  Description: Returns the next day of the specified weekday based on the specified date.

  ```
  SELECT next_day(timestamp'2019-09-09 00:00:00.000', 'monday');-- 2019-09-16 00:00:00.000
  SELECT next_day(date'2019-09-09', 'monday');-- 2019-09-16
  ```

- numtoday(integer) -> BIGINT

  Description: Converts transferred integer values to values of the day type, for example, BIGINT.

  ```
  SELECT numtoday(2);-- 2
  ```

11.13.6.14 Aggregate Functions

The aggregate function operates on a set of values to obtain a single value.

Except **count()**, **count_if()**, **max_by()**, **min_by()**, and **approx_distinct()**, other aggregate functions ignore null values and return null values when there is no

input row or all values are null. For example, **sum()** returns null instead of 0, and **avg()** does not take null values during statistics collection. The **coalesce** function converts null to 0.

Clause of an Aggregate Function

- Order by

  Some aggregate functions may generate different results due to different sequences of input values. You can use the **order by** clause in the aggregate function to specify the sequence.

  ```
  array_agg(x ORDER BY y DESC);
  array_agg(x ORDER BYx,y,z);
  ```

- Filter

  You can use the **filter** keyword to filter out unnecessary rows by using the **where** condition expression during aggregation. All aggregate functions support this function.

  ```
  aggregate_function(...) FILTER (WHERE <condition>)
  ```

Example:

```
--Create a table.
create table fruit (name varchar, price int);
--Insert data.
insert into fruit values ('peach',5),('apple',2);
--Sorting:
select array_agg (name order by price) from fruit;-- [apple, peach]
--Filtering:
select array_agg(name) filter (where price<10) from fruit;-- [peach, apple]
```

Common Aggregate Functions

An aggregate function usually applies to a specific field in a data set (table or view). The following parameter *x* is used to refer to the field.

- arbitrary(x)

  Description: Returns a non-null value of X. The return type is the same as X.

  ```
  select arbitrary(price) from fruit;-- 5
  ```

- array_agg(x)

  Description: Returns an array of input x fields of the same type as the input field.

  ```
  select array_agg(price) from fruit;-- [5,2]
  ```

- avg(x)

  Description: Returns the average of all input values in **double** type.

  ```
  select avg(price) from fruit;-- 3.5
  ```

- avg(time interval type)

  Description: Returns the average length of all input intervals. The return type is **interval**.

  ```
  select avg(last_login) from (values ('admin',interval '0 06:15:30' day to second),('user1',interval '0 07:15:30' day to second),('user2',interval '0 08:15:30' day to second)) as login_log(user,last_login);
  -- 0 07:15:30.000  Assume that a log table records the time since the last login. The result indicates that the average login interval is 0 days, 7 hours, 15 minutes, and 30 seconds.
  ```

- bool_and(boolean value)

  Description: Returns true if each input value is **true**, otherwise returns **false**.


```
select bool_and(isfruit) from (values ('item1',true), ('item2',false),('item3',true)) as
items(item,isfruit);--false
select bool_and(isfruit) from (values ('item1',true), ('item2',true),('item3',true)) as
items(item,isfruit);-- true
```

- bool_or(boolean value)

  Description: Returns **true** if any of the input values is **true**, otherwise returns **false**.

  ```
  select bool_or(isfruit) from (values ('item1',false), ('item2',false),('item3',false)) as
  items(item,isfruit);-- false
  select bool_or(isfruit) from (values ('item1',true), ('item2',false),('item3',false)) as items(item,isfruit); --
  true
  ```

- checksum(x)

  Description: Returns the checksum of the input value, which is not affected by the input sequence. The result type is **varbinary**.

  ```
  select checksum(price) from fruit; -- fb 28 f3 9a 9a fb bf 86
  ```

- count(*)

  Description: Returns the number of input records. The result type is **bigint**.

  ```
  select count(*) from fruit; -- 2
  ```

- count(x)

  Description: Returns the number of records whose input field is not null. The result type is **bigint**.

  ```
  select count(name) from fruit;-- 2
  ```

- count_if(x)

  Description: Returns the number of records whose input value is **true**. This function is similar to count**(CASE WHEN x THEN 1 END)** and is of the **bigint** type.

  ```
  select count_if(price>7) from fruit;-- 0
  ```

- every(boolean)

  Description: Is an alias for **bool_and()**.

- geometric_mean(x)

  Description: Returns the geometric mean of the input field value. The value is of the **double** type.

  ```
  select geometric_mean(price) from fruit; -- 3.162277660168379
  ```

- listagg(x, separator) → varchar

  Description: Returns a string concatenated by input values separated by specified separators.

  Syntax:

  ```
  LISTAGG( expression [, separator] [ON OVERFLOW overflow_behaviour])    WITHIN GROUP (ORDER
  BY sort_item, [...])
  ```

  If separator is not specified, the null character is used as the separator by default.

  ```
  SELECT listagg(value, ',') WITHIN GROUP (ORDER BY value) csv_value FROM (VALUES 'a', 'c', 'b')
  t(value);
  csv_value
  -----------
  'a,b,c'
  ```

  When the return value of this function exceeds 1048576 bytes, you can use **overflow_behaviour** to specify the action to take in this case. By default, an error will be thrown.


```
SELECT listagg(value, ',' ON OVERFLOW ERROR) WITHIN GROUP (ORDER BY value) csv_value FROM (VALUES 'a', 'b', 'c') t(value);
```

If the return value exceeds 1048576 bytes, truncate the extra non-null string and replace it with the string specified by **TRUNCATE**. **WITH COUNT** and **WITHOUT COUNT** indicate whether the return value contains the count.

```
SELECT LISTAGG(value, ',' ON OVERFLOW TRUNCATE '.....' WITH COUNT) WITHIN GROUP (ORDER BY value)FROM (VALUES 'a', 'b', 'c') t(value);
```

The **listagg** function can also be used for grouping strings. The following is an example:

```
SELECT id, LISTAGG(value, ',') WITHIN GROUP (ORDER BY o) csv_value FROM (VALUES
    (100, 1, 'a'),
    (200, 3, 'c'),
    (200, 2, 'b') ) t(id, o, value)
GROUP BY id
ORDER BY id;
id   | csv_value
-----+-------------
100  | a
200  | b,c
```

- max_by(x, y)

  Description: Returns the value of **x** associated with the maximum value of the **y** field in all input values.

  ```
  select max_by(name,price) from fruit; -- peach
  ```

- max_by(x, y, n)

  Description: Returns *n* **x** values sorted by **y** in descending order.

  ```
  select max_by(name,price,2) from fruit;-- [peach, apple]
  ```

- min_by(x,y)

  Description: Returns the value of **x** associated with the minimum value of the **y** field in all input values.

  ```
  select min_by(name,price) from fruit;-- apple
  ```

- min_by(x, y, n)

  Description: Returns *n* **x** values sorted by **y** in ascending order.

  ```
  select min_by(name,price,2) from fruit;-- [apple, peach]
  ```

- max(x)

  Description: Returns the maximum value of the input field **x**.

  ```
  select max(price) from fruit;-- 5
  ```

- max(x, n)

  Description: Returns the first *n* values of the input field **x** in descending order.

  ```
  select max(price,2) from fruit; -- [5, 2]
  ```

- min(x)

  Description: Returns the minimum value of the input field **x**.

  ```
  select min(price) from fruit;-- 2
  ```

- min(x, n)

  Description: Returns the first *n* values of the input field **x** in ascending order.

  ```
  select min(price,2) from fruit;-- [2, 5]
  ```

- sum(T, x)

  Description: Sums up the input field *x*. *T* is of the numeric type, for example, **int**, **double**, or **interval day to second**.

  ```
  select sum(price) from fruit;-- 7
  ```


- regr_avgx(T independent, T dependent) → double

  Description: Calculates the average value of the independent variable (*expr2*) of the regression line. After the empty pair (*expr1, expr2*) is removed, the value is *AVG(expr2)*.

```
create table sample_collection(id int,time_cost int,weight decimal(5,2));

insert into sample_collection values
(1,5,86.38),
(2,10,281.17),
(3,15,89.91),
(4,20,17.5),
(5,25,88.76),
(6,30,83.94),
(7,35,44.26),
(8,40,17.4),
(9,45,5.6),
(10,50,145.68);

select regr_avgx(time_cost,weight) from sample_collection;
       _col0
--------------------
 86.06000000000002
(1 row)
```

- regr_avgy(T independent, T dependent) → double

  Description: Calculates the average value of the dependent variable (*expr1*) of the regression line. After the empty pair (*expr1, expr2*) is removed, the value is *AVG(expr1)*.

```
select regr_avgy(time_cost,weight) from sample_collection;
 _col0
-------
  27.5
(1 row)
```

- regr_count(T independent, T dependent) → double

  Description: Returns the non-null logarithm used to fit a linear regression line.

```
select regr_count(time_cost,weight) from sample_collection;
 _col0
-------
    10
(1 row)
```

- regr_r2(T independent, T dependent) → double

  Description: Returns the coefficient of determination for regression.

```
select regr_r2(time_cost,weight) from sample_collection;
       _col0
--------------------
 0.1446739237728169
(1 row)
```

- regr_sxx(T independent, T dependent) → double

  Description: Returns the value of REGR_COUNT(expr1, expr2) * VAR_POP(expr2).

```
select regr_sxx(time_cost,weight) from sample_collection;
       _col0
--------------------
 59284.886600000005
(1 row)
```

- regr_sxy(T independent, T dependent) → double

  Description: Returns the value of REGR_COUNT(expr1, expr2) * COVAR_POP(expr1, expr2).


```
select regr_sxy(time_cost,weight) from sample_collection;
 _col0
----------
 -4205.95
(1 row)
```

- regr_syy(T independent, T dependent) → double

  Description: Returns the value of REGR_COUNT(expr1, expr2) * VAR_POP(expr1).

```
select regr_syy(time_cost,weight) from sample_collection;
 _col0
--------
 2062.5
(1 row)
```

Bitwise Aggregate Function

- bitwise_and_agg(x)

  Description: Uses two's complement to represent the bitwise AND operation on the input field **x**. The return type is **bigint**.

```
select bitwise_and_agg(x) from (values (31),(32)) as t(x);-- 0
```

- bitwise_or_agg(x)

  Description: Uses two's complement to represent the bitwise OR operation on the input field **x**. The return type is **bigint**.

```
select bitwise_or_agg(x) from (values (31),(32)) as t(x);-- 63
```

Map Aggregate Function

- histogram(x) -> map(K, bigint)

  Description: Returns a map containing the number of occurrences of all input field **x**.

```
select histogram(x),histogram(y) from (values (15,17),(15,18),(15,19),(15,20)) as t(x,y);-- {15=4}, {17=1, 18=1, 19=1, 20=1}
```

- map_agg(key, value) -> map(K, V)

  Description: Returns a map whose input field key and input field value are key-value pairs.

```
select map_agg(name,price) from fruit;-- {apple=2, peach=5}
```

- map_union(x(K, V)) -> map(K, V)

  Description: Returns the union of all input maps. If a key appears multiple times in the input set, the corresponding value is any value corresponding to the key in the input set.

```
select map_union(x) from (values (map(array['banana'],array[10.0])),
(map(array['apple'],array[7.0]))) as t(x);-- {banana=10.0, apple=7.0}
select map_union(x) from (values (map(array['banana'],array[10.0])),
(map(array['banana'],array[7.0]))) as t(x);-- {banana=10.0}
```

- multimap_agg(key, value) -> map(K, array(V))

  Description: Returns a map consisting of input key-value pairs. Each key can correspond to multiple values.

```
select multimap_agg(key, value) from (values ('apple',7),('apple',8),('apple',8),('lemon',5) ) as t(key,value); - {apple=[7, 8, 8], lemon=[5]}
```

Approximation Aggregate Function

In actual situations, when we collect statistics on a large amount of data, sometimes we only care about an approximate value instead of a specific value.

For example, when we collect statistics on the sales volume of a product, the approximation aggregate function is useful because it uses less memory and CPU resources, in this way, you can obtain data results without any problems, such as overflowing to disks or CPU peaks. This is useful for billions of rows of data calculations.

- approx_median(x) → bigint

  Description: Calculates the median value of a distribution of values.

  ```
  select approx_median(price) from fruit; -- 10.0
  ```

- approx_distinct(x) → bigint

  Description: The return type of this function is **bigint**, which provides an approximate count of **count(distinct x)**. If all inputs are null, **0** is returned.

  The errors of all possible values of this function relative to the correct values follow an approximate normal distribution with a standard deviation of 2.3%. It does not guarantee the upper limit of any specific input set error.

  ```
  select approx_distinct(price) from fruit; -- 2
  ```

- approx_distinct(x, e)→ bigint

  Description: The return type of this function is **bigint**, which provides an approximate count of **count(distinct x)**. If all inputs are null, **0** is returned.

  The errors of all possible values of this function relative to the correct values follow an approximate normal distribution with a standard deviation of less than *e*. It does not guarantee the upper limit of any specific input set error.

  In the current implementation of the function, the value range of e is [0.0040625, 0.26000].

  ```
  select approx_distinct(weight,0.0040625) from sample_collection; -- 10
  select approx_distinct(weight,0.26) from sample_collection; -- 8
  ```

- approx_most_frequent(*buckets*, *value*, *capacity*) → map<[same as value], bigint>

  Description: Approximately collects statistics on the top **buckets** elements that appear most frequently. Less memory will be used for approximate estimation of high-frequency values. A larger **capacity** value indicates a more accurate result, which consumes more memory. The return value of this function is a map that consists of key-value pairs of high-frequency values and their frequencies.

  ```
  SELECT approx_most_frequent(3, x, 15) FROM (values 'A', 'B', 'A', 'C', 'A', 'B', 'C', 'D', 'E') t(x); --  {A=3, B=2, C=2}
  SELECT approx_most_frequent(3, x, 100) FROM (values 1, 2, 1, 3, 1, 2, 3, 4, 5) t(x); -- {1=3, 2=2, 3=2}
  ```

  NOTE

  The commonly used quantiles are binary, quaternary, decimal, and percentile. This means that the input set is evenly divided into equal parts, and then the value at the corresponding position is found. For example, **approx_percentile(x, 0.5)** is used to find a value that is about 50% of the x value, that is, a 2-quantile value.

- approx_percentile(x, percentage)→ [same as x]

  Description: Returns the approximate percentile based on the given percentage. The percentage value must be a constant between 0 and 1 for all input rows.

  ```
  select approx_percentile(x, 0.5) from ( values (2),(3),(7),(8),(9)) as t(x); --7
  ```

- approx_percentile(x, percentages)→ array<[same as x]>

Description: Returns an approximate percentile of the x value of all input fields in a given percentage array. Each value in this percentage array must be a constant between 0 and 1 for all input rows.

```
select approx_percentile(x, array[0.1,0.2,0.3,0.5]) from ( values (2),(3),(7),(8),(9)) as t(x); --[2, 3, 3, 7]
```

- approx_percentile(x, w, percentage) → array<[same as x]>

  Description: Returns the approximate percentile of all **x** input values by **percentage**. The weight of each item is **w** and must be a positive number. Set a valid percentile for **x**. The value of **percentage** must range from 0 to 1, and all input rows must be constants.

  ```
  select approx_percentile(x, 5,array[0.1,0.2,0.3,0.5]) from ( values (2),(3),(7),(8),(9)) as t(x); --[2, 3, 3, 7]
  ```

- approx_percentile(x, w, percentage, accuracy) → [same as x]

  Description: Returns the approximate percentile of all **x** input values by **percentage**. The weight of each item is **w** and must be a positive number. Set a valid percentile for **x**. The value of **percentage** must range from 0 to 1, and all input rows must be constants. The maximum progress error of the approximate value is specified by **accuracy**.

  ```
  select approx_percentile(x, 5,0.5,0.97) from ( values (2),(3),(7),(8),(9)) as t(x); --7
  ```

- approx_percentile(x, w, percentages) → [same as x]

  Description: Returns an approximate percentile of all **x** input values based on each percentage value in the percentage array. The weight of each item is **w** and must be a positive number. Set a valid percentile for **x**. Each element value in the percentage array must range from 0 to 1, and all input rows must be constants.

  ```
  select approx_percentile(x,5, array[0.1,0.2,0.3,0.5]) from ( values (2),(3),(7),(8),(9)) as t(x);  -- [2, 3, 3, 7]
  ```

  NOTE

  The preceding **approx_percentile** function also supports the **percentile_approx** function of the same parameter set.

- numeric_histogram(buckets, value, weight)

  Description: Calculates the approximate histogram for all values based on the number of buckets. The width of each item uses **weight**. This algorithm is based on:

  ```
  Yael Ben-Haim and Elad Tom-Tov, "A streaming parallel decision tree algorithm", J. Machine Learning Research 11 (2010), pp. 849--872.
  ```

  The value of **buckets** must be **bigint**. The values of **value** and **weight** must be numbers.

  ```
  select numeric_histogram(20,x,4) from ( values (2),(3),(7),(8),(9)) as t(x);
  _col0
  ------------------------------------------------
   {2.0=4.0, 3.0=4.0, 7.0=4.0, 8.0=4.0, 9.0=4.0}
  (1 row)
  ```

- numeric_histogram(buckets, value)

  Description: Compared with **numeric_histogram(buckets, value,weight)**, this function sets **weight** to **1**.

  ```
  select numeric_histogram(20,x) from ( values (2),(3),(7),(8),(9)) as t(x);
  _col0
  ------------------------------------------------
   {2.0=1.0, 3.0=1.0, 7.0=1.0, 8.0=1.0, 9.0=1.0}
  (1 row)
  ```


Statistical Aggregate Function

- corr(y,x)

  Description: Returns the correlation coefficient of the input value.

  ```
  select corr(y,x) from (values (1,5),(2,6),(3,7),(4,8)) as t(x,y);-- 1.0
  ```

- covar_pop(y, x)

  Description: Returns the population covariance of the input value.

  ```
  select covar_pop(y,x) from (values (1,5),(2,6),(3,7),(4,8)) as t(x,y); --1.25
  ```

- *covar_samp(y, x)*

  Description: Returns the sample covariance of the input value.

  ```
  select covar_samp(y,x) from (values (1,5),(2,6),(3,7),(4,8)) as t(x,y);-- 1.6666666
  ```

- kurtosis(*x*)

  Description: Kurtosis, also called peak-state coefficient, indicates the number of peak values at the average value of the probability density distribution curve, that is, the statistical value that describes the steepness of all value distribution forms in the entire system. Intuitively, kurtosis reflects the sharpness of the peak. This statistic needs to be compared with the normal distribution.

  The kurtosis is defined as the **4th standardized central moment** of the sample.

  The kurtosis of a random variable is the ratio of the fourth-order central moment of the random variable to the square of the variance.

  The calculation formula is as follows:

  $$\text{Kurtosis} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^4 / \text{SD}^4 - 3$$

  ```
  select kurtosis(x) from (values (1),(2),(3),(4)) as t(x); -- -1.1999999999999993
  ```

- regr_intercept(y, x)

  Description: Returns the linear regression intercept of the input value. **y** is a subordinate value. **x** is an independent value.

  ```
  select regr_intercept(y,x) from (values (1,5),(2,6),(3,7),(4,8)) as t(x,y);-- 4.0
  ```

- regr_slope(y, x)

  Description: Returns the linear regression slope of the input value. **y** is a subordinate value. **x** is an independent value.

  ```
  select regr_slope(y,x) from (values (1,5),(2,6),(3,7),(4,8)) as t(x,y);--  1.0
  ```

- skewness(x)

  Description: Returns the skew degree of all input values.

  ```
  select skewness(x) from (values (1),(2),(3),(4)) as t(x); -- 0.0
  ```

- stddev(x)

  Description: Alias of **stedev_samp()**

- stddev_pop(x)

  Description: Returns the population standard deviation of all input values.

  ```
  select stddev_pop(x) from (values (1),(2),(3),(4)) as t(x);--  1.118033988749895
  ```

- stddev_samp(x)

  Description: Returns the sample standard deviation of all input values.


```
select stddev_samp(x) from (values (1),(2),(3),(4)) as t(x);--  1.2909944487358056
```

- variance(x)

  Description: Alias of **var_samp()**

- var_pop(x)

  Description: Returns the population variance of all input values.

  ```
  select var_pop(x) from (values (1),(2),(3),(4)) as t(x);-- 1.25
  ```

- var_samp(x)

  Description: Returns the sample variance of all input values.

  ```
  select var_samp(x) from (values (1),(2),(3),(4)) as t(x);--  1.6666666666666667
  ```

Lambda Aggregation Function

reduce_agg(*inputValue T*, *initialState S*, *inputFunction(S*, *T*, *S)*, *combineFunction(S*, *S*, *S)*)

inputFunction is called for each non-null input value. In addition to obtaining the input value, **inputFunction** also obtains the current status, which is **initialState** initially, and then returns the new status. **CombineFunction** is invoked to combine the two states into a new state. Return the final status:

```
SELECT id, reduce_agg(value, 0, (a, b) -> a + b, (a, b) -> a + b)
FROM (
   VALUES
      (1, 3),
      (1, 4),
      (1, 5),
      (2, 6),
      (2, 7)
) AS t(id, value)
GROUP BY id;
-- (1, 12)
-- (2, 13)

SELECT id, reduce_agg(value, 1, (a, b) -> a * b, (a, b) -> a * b)
FROM (
   VALUES
      (1, 3),
      (1, 4),
      (1, 5),
      (2, 6),
      (2, 7)
) AS t(id, value)
GROUP BY id;
-- (1, 60)
-- (2, 42)
```

NOTE

The status value must be Boolean, integer, floating-point, date, time, or interval.

11.13.6.15 Window Functions

Window functions perform calculations across rows of query results. They are run after the **HAVING** clause but before the **ORDER BY** clause. To call a window function, you need to use the **OVER** clause to specify the special syntax of the window. The window consists of three parts:

- Partition specification, which divides the input rows into different partitions. This is similar to how the **GROUP BY** clause divides rows into different groups in an aggregate function.

- Sorting specification, which determines the order in which the window function will process the input rows
- Window frame, which specifies the sliding window of the row to be processed by the specified function. If no frame is specified, the default value is **RANGE UNBOUNDED PRECEDING**, which is the same as the value of **UNBOUNDEEN PREBODING AND CURRENT ROWGE**. The frame contains all rows from the beginning of the partition to the last peer of the current row. In the absence of **ORDER BY**, all rows are treated as equivalent, so the range between the unbound preamble and the current row is equal to the range between the unbound preamble and the unbound subsequent rows.

  For example, in the following query, the information in the **salary** table is sorted according to the salary of employees in each department.

```
-- Create a data table and insert data.
create table salary (dept varchar, userid varchar, sal double);
 insert into salary values ('d1','user1',1000),('d1','user2',2000),('d1','user3',3000),('d2','user4',4000),
('d2','user5',5000);

--Query data.
select dept,userid,sal,rank() over (partition by dept order by sal desc) as rnk from salary order by
dept,rnk;
dept | userid |  sal   | rnk
------|--------|--------|-----
 d1   | user3  | 3000.0 |   1
 d1   | user2  | 2000.0 |   2
 d1   | user1  | 1000.0 |   3
 d2   | user5  | 5000.0 |   1
 d2   | user4  | 4000.0 |   2
```

Aggregate Functions

All aggregate functions can be used as window functions by adding **over** clauses. The aggregate function operates on each row of records in the current window framework.

The following query generates the rolling sum of the order price calculated by each employee by day:

```
select dept,userid,sal,sum(sal) over (partition by dept order by sal desc) as rolling_sum from salary order by
dept,userid,sal;
dept | userid |  sal   | rolling_sum
------|--------|--------|-------------
 d1   | user1  | 1000.0 |      6000.0
 d1   | user2  | 2000.0 |      5000.0
 d1   | user3  | 3000.0 |      3000.0
 d2   | user4  | 4000.0 |      9000.0
 d2   | user5  | 5000.0 |      5000.0
(5 rows)
```

Ranking Functions

- cume_dist() → bigint

  Description: Number of lines less than or equal to the current value/Total number of lines in the group – For example, calculate the proportion of the number of employees whose salary is less than or equal to the current salary to the total number of employees.

```
 --Query Example
SELECT dept, userid, sal, CUME_DIST() OVER(ORDER BY sal) AS rn1, CUME_DIST() OVER(PARTITION
BY dept ORDER BY sal) AS rn2 FROM salary;
dept | userid |  sal   | rn1 |        rn2
------|--------|--------|-----|--------------------
```



```
 d2   | user4  | 4000.0 | 0.8 |                0.5
 d2   | user5  | 5000.0 | 1.0 |                1.0
 d1   | user1  | 1000.0 | 0.2 | 0.3333333333333333
 d1   | user2  | 2000.0 | 0.4 | 0.6666666666666666
 d1   | user3  | 3000.0 | 0.6 |                1.0
(5 rows)
```

- dense_rank() → bigint

  Description: Ranking of the returned value in a group of values. This is similar to **rank ()**. The difference is that the tie value does not generate gaps in the sequence.

- ntile(*n*) → bigint

  Description: Divides packet data into n fragments in sequence and returns the current fragment value. NTILE does not support ROWS BETWEEN. For example, if **NTILE(2) OVER(PARTITION BY cookieid ORDER BY createtime ROWS BETWEEN 3 PRECEDING AND CURRENT ROW)** fragments are not evenly distributed, the distribution of the first fragment is added by default.

```
--Create a table and insert data into the table.
create table cookies_log (cookieid varchar,createtime  date,pv int);
insert into cookies_log values
    ('cookie1',date '2020-07-10',1),
    ('cookie1',date '2020-07-11',5),
    ('cookie1',date '2020-07-12',7),
    ('cookie1',date '2020-07-13',3),
    ('cookie1',date '2020-07-14',2),
    ('cookie1',date '2020-07-15',4),
    ('cookie1',date '2020-07-16',4),
    ('cookie2',date '2020-07-10',2),
    ('cookie2',date '2020-07-11',3),
    ('cookie2',date '2020-07-12',5),
        ('cookie2',date '2020-07-13',6),
        ('cookie2',date '2020-07-14',3),
        ('cookie2',date '2020-07-15',9),
    ('cookie2',date '2020-07-16',7);
-- Query results.
SELECT cookieid,createtime,pv,
NTILE(2) OVER(PARTITION BY cookieid ORDER BY createtime) AS rn1,-- divides data into two
fragments in a group.
NTILE(3) OVER(PARTITION BY cookieid ORDER BY createtime) AS rn2,  -- divides data into three
fragments in a group.
NTILE(4) OVER(ORDER BY createtime) AS rn3   --divides all data into four fragments.
FROM cookies_log
ORDER BY cookieid,createtime;
cookieid  | createtime | pv | rn1 | rn2 | rn3
----------|------------|----|-----|-----|-----
 cookie1  | 2020-07-10 |  1 |   1 |   1 |   1
 cookie1  | 2020-07-11 |  5 |   1 |   1 |   1
 cookie1  | 2020-07-12 |  7 |   1 |   1 |   2
 cookie1  | 2020-07-13 |  3 |   1 |   2 |   2
 cookie1  | 2020-07-14 |  2 |   2 |   2 |   3
 cookie1  | 2020-07-15 |  4 |   2 |   3 |   4
 cookie1  | 2020-07-16 |  4 |   2 |   3 |   4
 cookie2  | 2020-07-10 |  2 |   1 |   1 |   1
 cookie2  | 2020-07-11 |  3 |   1 |   1 |   1
 cookie2  | 2020-07-12 |  5 |   1 |   1 |   2
 cookie2  | 2020-07-13 |  6 |   1 |   2 |   2
 cookie2  | 2020-07-14 |  3 |   2 |   2 |   3
 cookie2  | 2020-07-15 |  9 |   2 |   3 |   3
 cookie2  | 2020-07-16 |  7 |   2 |   3 |   4
(14 rows)
```

- percent_rank() → double

  Description: Rankings of return values in percentage within a set of values. The result is *(r-1)/(n-1)*, where r is the **rank ()** of the row and *n* is the total number of rows in the window partition.


```
SELECT dept,userid,sal,
PERCENT_RANK() OVER(ORDER BY sal) AS rn1, -- in a group
RANK() OVER(ORDER BY sal) AS rn11,          -- RANK value in a group
SUM(1) OVER(PARTITION BY NULL) AS rn12,     --Total number of lines in the group
PERCENT_RANK() OVER(PARTITION BY dept ORDER BY sal) AS rn2
from salary;
 dept | userid |  sal   | rn1  | rn11 | rn12 | rn2
------|--------|--------|------|------|------|-----
 d2   | user4  | 4000.0 | 0.75 |    4 |    5 | 0.0
 d2   | user5  | 5000.0 |  1.0 |    5 |    5 | 1.0
 d1   | user1  | 1000.0 |  0.0 |    1 |    5 | 0.0
 d1   | user2  | 2000.0 | 0.25 |    2 |    5 | 0.5
 d1   | user3  | 3000.0 |  0.5 |    3 |    5 | 1.0
(5 rows)
```

- rank() → bigint

  Description: Ranking of the returned value in a group of values. The level is 1 plus the number of rows that are not equal to the current row. Therefore, the average value in the sort will create a gap in the sequence. Ranks each window partition.

```
SELECT
cookieid,
createtime,
pv,
RANK() OVER(PARTITION BY cookieid ORDER BY pv desc) AS rn1,
DENSE_RANK() OVER(PARTITION BY cookieid ORDER BY pv desc) AS rn2,
ROW_NUMBER() OVER(PARTITION BY cookieid ORDER BY pv DESC) AS rn3
FROM cookies_log
WHERE cookieid = 'cookie1';
 cookieid | createtime | pv | rn1 | rn2 | rn3
----------|------------|----|-----|-----|-----
 cookie1  | 2020-07-12 |  7 |   1 |   1 |   1
 cookie1  | 2020-07-11 |  5 |   2 |   2 |   2
 cookie1  | 2020-07-15 |  4 |   3 |   3 |   3
 cookie1  | 2020-07-16 |  4 |   3 |   3 |   4
 cookie1  | 2020-07-13 |  3 |   5 |   4 |   5
 cookie1  | 2020-07-14 |  2 |   6 |   5 |   6
 cookie1  | 2020-07-10 |  1 |   7 |   6 |   7
(7 rows)
```

- row_number() → bigint

  Description: Starting from 1, the sequence of records in a group is generated in sequence. For example, generate the daily **pv** ranking in descending order. There are many application scenarios for **ROW_NUMBER()**. For another example, obtain the record that ranks first in a group, or obtain the first **refer** in a session.

```
SELECT cookieid, createtime, pv, ROW_NUMBER() OVER(PARTITION BY cookieid ORDER BY pv desc)
AS rn from cookies_log;
  cookieid | createtime | pv | rn
 ----------|------------ |----|----
 cookie2  | 2020-07-15 |  9 |  1
 cookie2  | 2020-07-16 |  7 |  2
 cookie2  | 2020-07-13 |  6 |  3
 cookie2  | 2020-07-12 |  5 |  4
 cookie2  | 2020-07-14 |  3 |  5
 cookie2  | 2020-07-11 |  3 |  6
 cookie2  | 2020-07-10 |  2 |  7
 cookie1  | 2020-07-12 |  7 |  1
 cookie1  | 2020-07-11 |  5 |  2
 cookie1  | 2020-07-15 |  4 |  3
 cookie1  | 2020-07-16 |  4 |  4
 cookie1  | 2020-07-13 |  3 |  5
 cookie1  | 2020-07-14 |  2 |  6
 cookie1  | 2020-07-10 |  1 |  7
(14 rows)
```


Value Functions

Generally, the null value must be considered. If **IGNORE NULLS** is specified, all rows containing x whose value is null will be excluded. If the value of **x** in all rows is null, the default value is returned. Otherwise, null is returned.

```
Data Preparation
create table cookie_views( cookieid varchar,createtime timestamp,url varchar);
insert into cookie_views values
('cookie1',timestamp '2020-07-10 10:00:02','url20'),
('cookie1',timestamp '2020-07-10 10:00:00','url10'),
('cookie1',timestamp '2020-07-10 10:03:04','urll3'),
('cookie1',timestamp '2020-07-10 10:50:05','url60'),
('cookie1',timestamp '2020-07-10 11:00:00','url70'),
('cookie1',timestamp '2020-07-10 10:10:00','url40'),
('cookie1',timestamp '2020-07-10 10:50:01','url50'),
('cookie2',timestamp '2020-07-10 10:00:02','url23'),
('cookie2',timestamp '2020-07-10 10:00:00','url11'),
('cookie2',timestamp '2020-07-10 10:03:04','url33'),
('cookie2',timestamp '2020-07-10 10:50:05','url66'),
('cookie2',timestamp '2020-07-10 11:00:00','url77'),
('cookie2',timestamp '2020-07-10 10:10:00','url47'),
('cookie2',timestamp '2020-07-10 10:50:01','url55');
```

- first_value(*x*) → [same as input]

  Description: Returns the first value of the window.

  ```
  SELECT cookieid,
  createtime,
  url,
  ROW_NUMBER() OVER(PARTITION BY cookieid ORDER BY createtime) AS rn,
  FIRST_VALUE(url) OVER(PARTITION BY cookieid ORDER BY createtime) AS first1
  FROM cookie_views;
  cookieid |       createtime        |  url  | rn | first1
  ----------|-------------------------|-------|----|--------
   cookie1  | 2020-07-10 10:00:00.000 | url10 |  1 | url10
   cookie1  | 2020-07-10 10:00:02.000 | url20 |  2 | url10
   cookie1  | 2020-07-10 10:03:04.000 | urll3 |  3 | url10
   cookie1  | 2020-07-10 10:10:00.000 | url40 |  4 | url10
   cookie1  | 2020-07-10 10:50:01.000 | url50 |  5 | url10
   cookie1  | 2020-07-10 10:50:05.000 | url60 |  6 | url10
   cookie1  | 2020-07-10 11:00:00.000 | url70 |  7 | url10
   cookie2  | 2020-07-10 10:00:00.000 | url11 |  1 | url11
   cookie2  | 2020-07-10 10:00:02.000 | url23 |  2 | url11
   cookie2  | 2020-07-10 10:03:04.000 | url33 |  3 | url11
   cookie2  | 2020-07-10 10:10:00.000 | url47 |  4 | url11
   cookie2  | 2020-07-10 10:50:01.000 | url55 |  5 | url11
   cookie2  | 2020-07-10 10:50:05.000 | url66 |  6 | url11
   cookie2  | 2020-07-10 11:00:00.000 | url77 |  7 | url11
  (14 rows)
  ```

- last_value(*x*) → [same as input]

  Description: Returns the last value of the window.

  ```
  SELECT cookieid,createtime,url,
  ROW_NUMBER() OVER(PARTITION BY cookieid ORDER BY createtime) AS rn,
  LAST_VALUE(url) OVER(PARTITION BY cookieid ORDER BY createtime) AS last1
  FROM cookie_views;
   cookieid |       createtime        |  url  | rn | last1
  ----------|-------------------------|-------|----|-------
   cookie2  | 2020-07-10 10:00:00.000 | url11 |  1 | url11
   cookie2  | 2020-07-10 10:00:02.000 | url23 |  2 | url23
   cookie2  | 2020-07-10 10:03:04.000 | url33 |  3 | url33
   cookie2  | 2020-07-10 10:10:00.000 | url47 |  4 | url47
   cookie2  | 2020-07-10 10:50:01.000 | url55 |  5 | url55
   cookie2  | 2020-07-10 10:50:05.000 | url66 |  6 | url66
   cookie2  | 2020-07-10 11:00:00.000 | url77 |  7 | url77
   cookie1  | 2020-07-10 10:00:00.000 | url10 |  1 | url10
   cookie1  | 2020-07-10 10:00:02.000 | url20 |  2 | url20
  ```



```
 cookie1  | 2020-07-10 10:03:04.000 | urll3 |  3 | urll3
 cookie1  | 2020-07-10 10:10:00.000 | url40 |  4 | url40
 cookie1  | 2020-07-10 10:50:01.000 | url50 |  5 | url50
 cookie1  | 2020-07-10 10:50:05.000 | url60 |  6 | url60
 cookie1  | 2020-07-10 11:00:00.000 | url70 |  7 | url70
(14 rows)
```

- nth_value(*x*, *offset*) → [same as input]

  Description: Returns the value of the specified offset from the beginning of the window. The offset starts from 1. The offset can be any scalar expression. If the offset is null or greater than the number of values in the window, null is returned. The offset cannot be 0 or a negative number.

```
SELECT cookieid,createtime,url,
ROW_NUMBER() OVER(PARTITION BY cookieid ORDER BY createtime) AS rn,
NTH_VALUE(url,3) OVER(PARTITION BY cookieid ORDER BY createtime) AS last1
FROM cookie_views;
cookieid |       createtime        |  url  | rn | last1
----------|-------------------------|-------|----|-------
 cookie1  | 2020-07-10 10:00:00.000 | url10 |  1 | NULL
 cookie1  | 2020-07-10 10:00:02.000 | url20 |  2 | NULL
 cookie1  | 2020-07-10 10:03:04.000 | urll3 |  3 | urll3
 cookie1  | 2020-07-10 10:10:00.000 | url40 |  4 | urll3
 cookie1  | 2020-07-10 10:50:01.000 | url50 |  5 | urll3
 cookie1  | 2020-07-10 10:50:05.000 | url60 |  6 | urll3
 cookie1  | 2020-07-10 11:00:00.000 | url70 |  7 | urll3
 cookie2  | 2020-07-10 10:00:00.000 | url11 |  1 | NULL
 cookie2  | 2020-07-10 10:00:02.000 | url23 |  2 | NULL
 cookie2  | 2020-07-10 10:03:04.000 | url33 |  3 | url33
 cookie2  | 2020-07-10 10:10:00.000 | url47 |  4 | url33
 cookie2  | 2020-07-10 10:50:01.000 | url55 |  5 | url33
 cookie2  | 2020-07-10 10:50:05.000 | url66 |  6 | url33
 cookie2  | 2020-07-10 11:00:00.000 | url77 |  7 | url33
(14 rows)
```

- lead(*x*[, *offset*[, *default_value*]]) → [same as input]

  Description: Returns the value of the offset row after the current row in the window partition. The offset starts from 0, that is, the current line. The offset can be any scalar expression. The default offset is **1**. If the offset is null, null is returned. If the offset points to a row that is not in the partition, **default_value** is returned. If it is not specified, null is returned. The **lead()** function requires that the window sequence be specified. Do not specify a window frame.

```
SELECT cookieid,createtime,url,
ROW_NUMBER() OVER(PARTITION BY cookieid ORDER BY createtime) AS rn,
LEAD(createtime,1,timestamp '2020-01-01 00:00:00') OVER(PARTITION BY cookieid ORDER BY
createtime) AS next_1_time,
LEAD(createtime,2) OVER(PARTITION BY cookieid ORDER BY createtime) AS next_2_time
FROM cookie_views;
 cookieid |       createtime        |  url  | rn |       next_1_time       |       next_2_time
----------|-------------------------|-------|----|-------------------------|-------------------------
 cookie2  | 2020-07-10 10:00:00.000 | url11 |  1 | 2020-07-10 10:00:02.000 | 2020-07-10 10:03:04.000
 cookie2  | 2020-07-10 10:00:02.000 | url23 |  2 | 2020-07-10 10:03:04.000 | 2020-07-10 10:10:00.000
 cookie2  | 2020-07-10 10:03:04.000 | url33 |  3 | 2020-07-10 10:10:00.000 | 2020-07-10 10:50:01.000
 cookie2  | 2020-07-10 10:10:00.000 | url47 |  4 | 2020-07-10 10:50:01.000 | 2020-07-10 10:50:05.000
 cookie2  | 2020-07-10 10:50:01.000 | url55 |  5 | 2020-07-10 10:50:05.000 | 2020-07-10 11:00:00.000
 cookie2  | 2020-07-10 10:50:05.000 | url66 |  6 | 2020-07-10 11:00:00.000 | NULL
 cookie2  | 2020-07-10 11:00:00.000 | url77 |  7 | 2020-01-01 00:00:00.000 | NULL
 cookie1  | 2020-07-10 10:00:00.000 | url10 |  1 | 2020-07-10 10:00:02.000 | 2020-07-10 10:03:04.000
 cookie1  | 2020-07-10 10:00:02.000 | url20 |  2 | 2020-07-10 10:03:04.000 | 2020-07-10 10:10:00.000
 cookie1  | 2020-07-10 10:03:04.000 | urll3 |  3 | 2020-07-10 10:10:00.000 | 2020-07-10 10:50:01.000
 cookie1  | 2020-07-10 10:10:00.000 | url40 |  4 | 2020-07-10 10:50:01.000 | 2020-07-10 10:50:05.000
 cookie1  | 2020-07-10 10:50:01.000 | url50 |  5 | 2020-07-10 10:50:05.000 | 2020-07-10 11:00:00.000
 cookie1  | 2020-07-10 10:50:05.000 | url60 |  6 | 2020-07-10 11:00:00.000 | NULL
 cookie1  | 2020-07-10 11:00:00.000 | url70 |  7 | 2020-01-01 00:00:00.000 | NULL
(14 rows)
```


- lag(*x*[, *offset*[, *default_value*]]) → [same as input]

  Description: Returns the value of the offset row before the current row in the window partition. The offset starts from **0**, that is, the current row. The offset can be any scalar expression. The default offset is **1**. If the offset is null, null is returned. If the offset points to a row that is not in the partition, **default_value** is returned. If this parameter is not specified, **null** is returned. The **lag()** function requires that the window sequence be specified and the window frame cannot be specified.

```
 SELECT cookieid, createtime, url, ROW_NUMBER() OVER(PARTITION BY cookieid  ORDER BY
createtime) AS rn,
   LAG(createtime,1, timestamp '2020-01-01 00:00:00') OVER(PARTITION BY cookieid ORDER BY
createtime)  AS last_1_time,
   LAG(createtime,2) OVER(PARTITION BY cookieid ORDER BY createtime)  AS last_2_time
    FROM cookie_views;

cookieid |       createtime          |  url  | rn |       last_1_time       |       last_2_time
----------|-------------------------|-------|----|-------------------------|-----------------------
 cookie2  | 2020-07-10 10:00:00.000 | url11 |  1 | 2020-01-01 00:00:00.000 | NULL
 cookie2  | 2020-07-10 10:00:02.000 | url23 |  2 | 2020-07-10 10:00:00.000 | NULL
 cookie2  | 2020-07-10 10:03:04.000 | url33 |  3 | 2020-07-10 10:00:02.000 | 2020-07-10 10:00:00.000
 cookie2  | 2020-07-10 10:10:00.000 | url47 |  4 | 2020-07-10 10:03:04.000 | 2020-07-10 10:00:02.000
 cookie2  | 2020-07-10 10:50:01.000 | url55 |  5 | 2020-07-10 10:10:00.000 | 2020-07-10 10:03:04.000
 cookie2  | 2020-07-10 10:50:05.000 | url66 |  6 | 2020-07-10 10:50:01.000 | 2020-07-10 10:10:00.000
 cookie2  | 2020-07-10 11:00:00.000 | url77 |  7 | 2020-07-10 10:50:05.000 | 2020-07-10 10:50:01.000
 cookie1  | 2020-07-10 10:00:00.000 | url10 |  1 | 2020-01-01 00:00:00.000 | NULL
 cookie1  | 2020-07-10 10:00:02.000 | url20 |  2 | 2020-07-10 10:00:00.000 | NULL
 cookie1  | 2020-07-10 10:03:04.000 | urll3 |  3 | 2020-07-10 10:00:02.000 | 2020-07-10 10:00:00.000
 cookie1  | 2020-07-10 10:10:00.000 | url40 |  4 | 2020-07-10 10:03:04.000 | 2020-07-10 10:00:02.000
 cookie1  | 2020-07-10 10:50:01.000 | url50 |  5 | 2020-07-10 10:10:00.000 | 2020-07-10 10:03:04.000
 cookie1  | 2020-07-10 10:50:05.000 | url60 |  6 | 2020-07-10 10:50:01.000 | 2020-07-10 10:10:00.000
 cookie1  | 2020-07-10 11:00:00.000 | url70 |  7 | 2020-07-10 10:50:05.000 | 2020-07-10 10:50:01.000
(14 rows)
```

11.13.6.16 Array Functions and Operators

Subscript Operator: []

Description: Subscript operators are used to access elements in an array and create indexes starting from **1**.

```
select myarr[5] from (values array [1,4,6,78,8,9],array[2,4,6,8,10,12]) as t(myarr);
 _col0
-------
     8
    10
```

Concatenation Operator: ||

|| Operators are used to concatenate arrays or values of the same type.

```
SELECT ARRAY[1] || ARRAY[2];
 _col0
--------
 [1, 2]
(1 row)

SELECT ARRAY[1] || 2;
 _col0
--------
 [1, 2]
(1 row)

SELECT 2 || ARRAY[1];
```



```
 _col0
--------
 [2, 1]
(1 row)
```

Array Function

Subscript operator: []

The subscript operator **[]** is used to obtain the value of the corresponding position in the array.

```
SELECT ARRAY[5,3,41,6][1] AS first_element; -- 5
```

Concatenation Operator: ||

The || operator is used to concatenate an array with an array or an element of the same type:

```
SELECT ARRAY[1]||ARRAY[2];-- [1, 2]
SELECT ARRAY[1]||2; -- [1, 2]
SELECT 2||ARRAY[1];-- [2, 1]
```

- array_contains(*x*, *element*) → boolean

  Description: Returns **true** if the array **x** contains elements.

  ```
  select array_contains (array[1,2,3,34,4],4); -- true
  ```

- all_match(array(T), function(T, boolean)) → boolean

  Description: Returns whether all elements of an array satisfy the given assertion function. If all elements satisfy the assertion function or the array is empty, **true** is returned. If one or more elements do not satisfy the assertion function, **false** is returned. If the assertion function is not satisfied by one or more elements, it returns null, and the return value of the **all_match** function is also null.

  ```
  select all_match(a, x-> true) from (values array[]) t(a); -- true
  select all_match(a, x-> x>2) from (values array[4,5,7]) t(a); -- true
  select all_match(a, x-> x>2) from (values array[1,5,7]) t(a); -- false
  select all_match(a, x-> x>1) from (values array[NULL, NULL ,NULL]) t(a); --NULL
  ```

- any_match(array(T), function(T, boolean)) → boolean

  Description: Returns whether an array has elements that satisfy the assertion function. If one or more elements satisfy the assertion function, **true** is returned. If all elements cannot satisfy the assertion function or the array is empty, **false** is returned. If the assertion function is not satisfied by one or more elements, it returns null, and the return value of the **all_match** function is also null.

  ```
  select any_match(a, x-> true) from (values array[]) t(a); -- false
  select any_match(a, x-> x>2) from (values array[0,1,2]) t(a); -- false
  select any_match(a, x-> x>2) from (values array[1,5,7]) t(a); -- true
  select any_match(a, x-> x>1) from (values array[NULL, NULL ,NULL]) t(a); -- NULL
  ```

- array_distinct(*x*) → array

  Description: Outputs the deduplicated array **x**.

  ```
  select  array_distinct(array [1,1,1,1,1,1,3,3,3,33,4,5,6,6,6,6]);--  [1, 3, 33, 4, 5, 6]
  ```

- array_intersect(*x*, *y*) → array

  Description: Returns the intersection of two arrays after deduplication.

  ```
  select array_intersect(array [1,3,5,7,9],array [1,2,3,4,5]);
    _col0
  -----------
  ```



```
 [1, 3, 5]
(1 row)
```

- array_union(*x*, *y*) → array

  Description: Returns the union of two arrays.

```
select array_union(array [1,3,5,7,9],array [1,2,3,4,5]);
       _col0
-----------------------
 [1, 3, 5, 7, 9, 2, 4]
(1 row)
```

- array_except(*x*, *y*) → array

  Description: Returns an array of deduplicated elements that are in **x** but not in **y**.

```
select array_except(array [1,3,5,7,9],array [1,2,3,4,5]);
 _col0
--------
 [7, 9]
(1 row)
```

- array_join(*x*, *delimiter*, *null_replacement*) → varchar

  Description: Concatenates elements of a given array **x** with delimiters and replaces the null value in **x** with optional characters.

```
select array_join(array[1,2,3,null,5,6],'|','0');--  1|2|3|0|5|6
```

- array_max(*x*) → x

  Description: Returns the maximum value of array **x**.

```
select array_max(array[2,54,67,132,45]);  -- 132
```

- array_min(*x*) → x

  Description: Returns the minimum value of array **x**.

```
select array_min(array[2,54,67,132,45]);  -- 2
```

- array_position(x,element) → bigint

  Description: Returns the first occurrence position of an element in the array **x**. If the element is not found, **0** is returned.

```
select array_position(array[2,3,4,5,1,2,3],3); -- 2
```

- array_remove(*x*, *element*) → array

  Description: Removes elements whose values are elements from array **x** and returns the elements.

```
select array_remove(array[2,3,4,5,1,2,3],3); --  [2, 4, 5, 1, 2]
```

- array_sort(*x*) → array

  Sorts and returns the array **x**. The elements of **x** must be sortable. The empty element is placed at the end of the returned array.

```
select array_sort(array[2,3,4,5,1,2,3]); -- [1, 2, 2, 3, 3, 4, 5]
```

- array_sort(*array(T)*, *function(T*, *T*, *int))*

  Description: Sorts and returns the array based on the given comparator function. The comparator will use two parameters that can be null, representing the two elements of the array that can be null. When the first element that can be empty is less than, equal to, or greater than the second element that can be empty, **-1**, **0**, or **1** is returned. If the comparator function returns other values, including **null**, the query fails and an error is thrown.

```
SELECT array_sort(ARRAY [3, 2, 5, 1, 2], (x, y) -> IF(x < y, 1, IF(x = y, 0, -1)));
 _col0
--------
[5, 3, 2, 2, 1]
```



```
(1 row)

SELECT array_sort(ARRAY ['bc', 'ab', 'dc'], (x, y) -> IF(x < y, 1, IF(x = y, 0, -1)));
 _col0
--------
[dc, bc, ab]
(1 row)

-- The null value is placed in the front, and other values are sorted in descending order.
SELECT array_sort(ARRAY [3, 2, null, 5, null, 1, 2],
                  (x, y) -> CASE WHEN x IS NULL THEN -1
                                 WHEN y IS NULL THEN 1
                                 WHEN x < y THEN 1
                                 WHEN x = y THEN 0
                                 ELSE -1 END);
 _col0
--------
[null, null, 5, 3, 2, 2, 1]
(1 row)

-- The null value is placed in the back, and other values are sorted in descending order.
SELECT array_sort(ARRAY [3, 2, null, 5, null, 1, 2],
                  (x, y) -> CASE WHEN x IS NULL THEN 1
                                 WHEN y IS NULL THEN -1
                                 WHEN x < y THEN 1
                                 WHEN x = y THEN 0
                                 ELSE -1 END);
 _col0
--------
[5, 3, 2, 2, 1, null, null]
(1 row)

-- Sorting by character string length:
SELECT array_sort(ARRAY ['a', 'abcd', 'abc'],
                  (x, y) -> IF(length(x) < length(y), -1,
                              IF(length(x) = length(y), 0, 1)));
 _col0
--------
[a, abc, abcd]
(1 row)

-- Sorting by array length:
SELECT array_sort(ARRAY [ARRAY[2, 3, 1], ARRAY[4, 2, 1, 4], ARRAY[1, 2]],
                  (x, y) -> IF(cardinality(x) < cardinality(y),
                              -1,
                              IF(cardinality(x) = cardinality(y), 0, 1)));
 _col0
--------
[[1, 2], [2, 3, 1], [4, 2, 1, 4]]
(1 row)
```

- arrays_overlap(*x*, *y*)

  Description: Returns **true** if two arrays have common non-null elements. Otherwise, **false** is returned.

  ```
  select arrays_overlap(array[1,2,3],array[3,4,5]);-- true
  select arrays_overlap(array[1,2,3],array[4,5,6]);-- false
  ```

- cardinality(*x*)

  Description: Returns the capacity of array **x**.

  ```
  select cardinality(array[1,2,3,4,5,6]); --6
  ```

- concat(*array1*, *array2*, ..., *arrayN*)

  Description: Provides the same function as the SQL standard connection operator (||).

- combinations(*array(T)*, *n) -> array(array(T)*)

Description: Returns *n* element subgroups of the input array. If the input array has no duplicate items, the combination returns a subset of *n* elements:

```
SELECT combinations(ARRAY['foo', 'bar', 'baz'], 2); -- [[foo, bar], [foo, baz], [bar, baz]]

SELECT combinations(ARRAY[1, 2, 3], 2); -- [[1, 2], [1, 3], [2, 3]]

SELECT combinations(ARRAY[1, 2, 2], 2); -- [[1, 2], [1, 2], [2, 2]]
```

The subgroups and the elements in the subgroups, though not specified, are ordered. The value of **n** cannot be greater than 5, and the maximum number of generated subgroups cannot exceed 100000.

- contains(*x*, *element*)

  Description: Returns **true** if the array **x** contains elements.

  ```
  select contains(array[1,2,3,34,4],4); -- true
  ```

- element_at(*array(E)*, *index*)

  Description: Returns the elements of an array at the given index. If the value of *index* is greater than **0**, this function provides the same function as the SQL standard subscript operator (**[]**). However, when this function accesses an index whose length is greater than the array length, **null** is returned, and the subscript operator fails in this case. If the *index* is less than **0**, **element_at** accesses elements from the last to the first.

  ```
  select element_at(array['a','b','c','d','e'],3); -- c
  ```

- filter(*array(T)*, *function(T*, *boolean)) -> array(T*)

  Description: Deletes an array consisting of elements whose operation result is **true**.

  ```
  SELECT filter(ARRAY [], x -> true); -- []

  SELECT filter(ARRAY [5, -6, NULL, 7], x -> x > 0); -- [5, 7]

  SELECT filter(ARRAY [5, NULL, 7, NULL], x -> x IS NOT NULL); -- [5, 7]
  ```

- flatten(*x*)

  Description: Expands **array(array(T))** to **array(T)** in series.

- ngrams(*array(T)*, *n) -> array(array(T)*)

  Description: Returns the n-gram (subsequence of *n* adjacent elements) of an array. The sequence of n-grams in the result is not specified.

  ```
  SELECT ngrams(ARRAY['foo', 'bar', 'baz', 'foo'], 2); -- [[foo, bar], [bar, baz], [baz, foo]]

  SELECT ngrams(ARRAY['foo', 'bar', 'baz', 'foo'], 3); -- [[foo, bar, baz], [bar, baz, foo]]

  SELECT ngrams(ARRAY['foo', 'bar', 'baz', 'foo'], 4); -- [[foo, bar, baz, foo]]

  SELECT ngrams(ARRAY['foo', 'bar', 'baz', 'foo'], 5); -- [[foo, bar, baz, foo]]

  SELECT ngrams(ARRAY[1, 2, 3, 4], 2); -- [[1, 2], [2, 3], [3, 4]]
  ```

- none_match(*array(T)*, *function(T*, *boolean)*)

  Description: Returns whether an array has no elements matching the given predicate. If no element matches the predicate, **true** is returned (a special case is when the array is empty). The value is **false** if one or more elements match. The value is **null** if the predicate function returns **null** for one or more elements and **false** for all other elements.

- reduce(*array(T)*, *initialState S*, *inputFunction(S*, *T*, *S)*, *outputFunction(S*, *R)*)

  Returns a single value reduced from an array. **inputFunction** is called for each element in the array in sequence. In addition to getting the element,

inputFunction also gets the current state, initially **initialState**, and then returns the new state. **outputFunction** will be called to convert the final state to a result value. It may be a constant function (**i-> i**).

```
SELECT reduce(ARRAY [], 0, (s, x) -> s + x, s -> s); -- 0
SELECT reduce(ARRAY [5, 20, 50], 0, (s, x) -> s + x, s -> s); -- 75
SELECT reduce(ARRAY [5, 20, NULL, 50], 0, (s, x) -> s + x, s -> s); -- NULL
SELECT reduce(ARRAY [5, 20, NULL, 50], 0, (s, x) -> s + COALESCE(x, 0), s -> s); -- 75
SELECT reduce(ARRAY [5, 20, NULL, 50], 0, (s, x) -> IF(x IS NULL, s, s + x), s -> s); -- 75
SELECT reduce(ARRAY [2147483647, 1], CAST (0 AS BIGINT), (s, x) -> s + x, s -> s); -- 2147483648
SELECT reduce(ARRAY [5, 6, 10, 20],            CAST(ROW(0.0, 0) AS ROW(sum DOUBLE, count
INTEGER)),

            (s, x) -> CAST(ROW(x + s.sum, s.count + 1) AS ROW(sum DOUBLE, count INTEGER)),
            s -> IF(s.count = 0, NULL, s.sum / s.count)); -- 10.25
```

- repeat(*element*, *count*)

  Description: Number of times that the element is repeatedly output. The value is filled in the array.

  ```
  select repeat(4,5);-- [4, 4, 4, 4, 4]
  ```

- reverse(*x*)

  Description: Populates the returned array with array elements in reverse order.

  ```
  select reverse(array[1,2,3,4,5]); --[5, 4, 3, 2, 1]
  ```

- sequence(start, *stop)*

  Description: Outputs an array from *start* to *stop*. If the value of *start* is not greater than that of *stop*, the value is incremented by 1 each time. Otherwise, the value is decremented by 1 each time.

  The data type of *start* and *stop* can also be date or timestamp, which increases or decreases by one day.

  ```
  select sequence(5,1);
       _col0
  -----------------
   [5, 4, 3, 2, 1]
  (1 row)

  select sequence(5,10);
         _col0
  ----------------------
   [5, 6, 7, 8, 9, 10]
  (1 row)
  ```

- sequence(*start*, *stop*, *step)*→ array

  Description: Outputs from *start* to *stop* with a *step* length.

  ```
  select sequence(1,30,5);--  [1, 6, 11, 16, 21, 26]
  ```

- shuffle(*x*)→ array

  Description: Gets a new array based on the given array randomization.

  ```
  select shuffle(array[1,2,3,4,5]);-- [1, 5, 4, 2, 3]
  select shuffle(array[1,2,3,4,5]);--  [2, 1, 3, 5, 4]
  ```

- size(*x*) → bigint

  Description: Returns the capacity of array **x**.

  ```
  select size(array[1,2,3,4,5,6]); --6
  ```

- slice(*x*, *start*, *length*) → array

  Description: Subset array x starts from the beginning of the index (if the *start* is negative, the array starts from the end) and its length is *length*.

  ```
  select slice(array[1,2,3,4,5,6],2,3);-- [2, 3, 4]
  ```


- sort_array(x)

  For details, see **array_sort(x)**.

- trim_array(*x*, *n*) → array

  Description: Deletes the last **n** elements in an array.

  ```
  SELECT trim_array(ARRAY[1, 2, 3, 4], 1); -- [1, 2, 3]
  SELECT trim_array(ARRAY[1, 2, 3, 4], 2); -- [1, 2]
  ```

- transform(*array(T)*, *function(T, U)) -> array(U*)

  Description: Returns an array that is the result of applying the function to each element of the array.

  ```
  SELECT transform(ARRAY [], x -> x + 1); -- []
  SELECT transform(ARRAY [5, 6], x -> x + 1); -- [6, 7]
  SELECT transform(ARRAY [5, NULL, 6], x -> COALESCE(x, 0) + 1); -- [6, 1, 7]
  SELECT transform(ARRAY ['x', 'abc', 'z'], x -> x || '0'); -- [x0, abc0, z0]
  SELECT transform(ARRAY [ARRAY [1, NULL, 2], ARRAY[3, NULL]], a -> filter(a, x -> x IS NOT NULL));
  -- [[1, 2], [3]]
  ```

- zip(*array1, array2[, ...]) -> array(row*)

  Description: Merges a given array into a single row array by element. The *M*th element of the *N*th independent variable is the *N*th field of the *M*th output element. If the parameter length is uneven, the missing value is filled with **null**.

  ```
  SELECT zip(ARRAY[1, 2], ARRAY['1b', null, '3b']); --  [{1, 1b}, {2, NULL}, {NULL, 3b}]
  ```

- zip_with(*array(T)*, *array(U)*, *function(T, U, R)) -> array(R*)

  Description: Combines two given arrays one by one into a single array using functions. If an array is short, a null value is added to the end of the function to match the length of the longer array.

  ```
  SELECT zip_with(ARRAY[1, 3, 5], ARRAY['a', 'b', 'c'], (x, y) -> (y, x)); -- [{a, 1}, {b, 3}, {c, 5}]

  SELECT zip_with(ARRAY[1, 2], ARRAY[3, 4], (x, y) -> x + y); -- [4, 6]

  SELECT zip_with(ARRAY['a', 'b', 'c'], ARRAY['d', 'e', 'f'], (x, y) -> concat(x, y)); -- [ad, be, cf]

  SELECT zip_with(ARRAY['a'], ARRAY['d', null, 'f'], (x, y) -> coalesce(x, y)); -- [a, null, f]
  ```

11.13.6.17 Map Functions and Operators

Subscript Operator: []

Description: The **[]** operator is used to retrieve a value corresponding to a given key from a mapping.

```
select age_map['li'] from (values (map(array['li','wang'],array[15,27]))) as table_age(age_map);-- 15
```

Map Functions

- cardinality(*x*)

  Description: Returns the cardinality of map *x*.

  ```
  select cardinality(map(array['num1','num2'],array[11,12]));-- 2
  ```

- element_at(*map(K, V)*, *key*)

  Description: Returns the value of *key* in a map. If the map does not contain the *key*, **null** is returned.

  ```
  select element_at(map(array['num1','num2'],array[11,12]),'num1'); --11
  select element_at(map(array['num1','num2'],array[11,12]),'num3');-- NULL
  ```


- map()

  Description: Returns an empty map.

  ```
  select map();-- {}
  ```

- map(*array(K)*, *array(V)) -> map(K*, *V*)

  Description: Returns a map based on the given key-value pair array. The **map_agg()** and **multimap_agg()** functions in the aggregate function can also be used to generate maps.

  ```
  SELECT map(ARRAY[1,3],ARRAY[2,4]);-- {1=2, 3=4}
  ```

- map_from_entries(*array(row(K*, *V))) -> map(K*, *V*)

  Description: Generates a map using a given array.

  ```
  SELECT map_from_entries(ARRAY[(1, 'x'), (2, 'y')]); --  {1=x, 2=y}
  ```

- multimap_from_entries(*array(row(K*, *V))) -> map(K*, *array(V)*)

  Description: Returns a composite map based on a given row array. Each key can correspond to multiple values.

  ```
  SELECT multimap_from_entries(ARRAY[(1, 'x'), (2, 'y'), (1, 'z')]); -- {1=[x, z], 2=[y]}
  ```

- map_entries(*map(K*, *V)) -> array(row(K*, *V)*)

  Description: Generates a row array using the given map.

  ```
  SELECT map_entries(MAP(ARRAY[1, 2], ARRAY['x', 'y'])); --  [{1, x}, {2, y}]
  ```

- map_concat(*map1(K*, *V)*, *map2(K*, *V)*, *...*, *mapN(K*, *V))*

  Description: Returns the union of all the given maps. If a key is found in multiple given maps, that key's value in the resulting map comes from the last one of those maps. In the following example, key a uses the value 10 of the last map.

  ```
  select map_concat(map(ARRAY['a','b'],ARRAY[1,2]),map(ARRAY['a', 'c'], ARRAY[10, 20]));
        _col0
  -------------------
   {a=10, b=2, c=20}
  (1 row)
  ```

- map_filter(*map(K*, *V)*, *function(K*, *V*, *boolean)) -> map(K*, *V*)

  Description: Constructs a new map using only the entry that maps the given function to **true**.

  ```
  SELECT map_filter(MAP(ARRAY[], ARRAY[]), (k, v) -> true); -- {}
  SELECT map_filter(MAP(ARRAY[10, 20, 30], ARRAY['a', NULL, 'c']), (k, v) -> v IS NOT NULL); -- {10=a, 30=c}
  SELECT map_filter(MAP(ARRAY['k1', 'k2', 'k3'], ARRAY[20, 3, 15]), (k, v) -> v > 10); -- {k3=15, k1=20}
  ```

- map_keys(*x(K*, *V)) -> array(K*)

  Description: Returns all arrays constructed by keys in a map.

  ```
  select map_keys(map(array['num1','num2'],array[11,12])); -- [num1, num2]
  ```

- map_values(*x(K*, *V)) -> array(V*)

  Description: Returns all value-constructed arrays in a map.

  ```
  select map_values(map(array['num1','num2'],array[11,12]));-- [11, 12]
  ```

- map_zip_with(*map(K*, *V1)*, *map(K*, *V2)*, *function(K*, *V1*, *V2*, *V3))*

  Description: Merges two given maps into a map by applying the function to a pair of values with the same key. For keys that appear only in one map, **null** is passed as the value for the missing key.

  ```
  SELECT map_zip_with(MAP(ARRAY[1, 2, 3], ARRAY['a', 'b', 'c']), -- {1 -> ad, 2 -> be, 3 -> cf}
                      MAP(ARRAY[1, 2, 3], ARRAY['d', 'e', 'f']),
                      (k, v1, v2) -> concat(v1, v2));
        _col0
  ```



```
--------------------
 {1=ad, 2=be, 3=cf}
(1 row)

SELECT map_zip_with(MAP(ARRAY['k1','k2'],ARRAY[1,2]),Map(ARRAY['K2','k3'],ARRAY[4,9]),(k,v1,v2)->(v1,v2));  -- {k3={NULL, 9}, k1={1, NULL}, k2={2, NULL}, K2={NULL, 4}}
                                    _col0
--------------------------------------------------------------------------------
{k3={NULL, 9}, k1={1, NULL}, k2={2, NULL}, K2={NULL, 4}}
(1 row)

 SELECT map_zip_with(MAP(ARRAY['a', 'b', 'c'], ARRAY[1, 8, 27]), -- {a -> a1, b -> b4, c -> c9}
                     MAP(ARRAY['a', 'b', 'c'], ARRAY[1, 2, 3]),
                     (k, v1, v2) -> k || CAST(v1/v2 AS VARCHAR));
      _col0
--------------------
 {a=a1, b=b4, c=c9}
(1 row)
```

- transform_keys(*map(K1, V)*, *function(K1, V, K2)) -> map(K2, V)*

  Description: For each entry in the map, map the key value K1 to the new key value K2 and keep the corresponding value unchanged.

```
SELECT transform_keys(MAP(ARRAY[], ARRAY[]), (k, v) -> k + 1); -- {}

SELECT transform_keys(MAP(ARRAY [1, 2, 3], ARRAY ['a', 'b', 'c']), (k, v) -> k + 1); -- {2=a, 3=b, 4=c}

SELECT transform_keys(MAP(ARRAY ['a', 'b', 'c'], ARRAY [1, 2, 3]), (k, v) -> v * v); -- {1=1, 9=3, 4=2}

SELECT transform_keys(MAP(ARRAY ['a', 'b'], ARRAY [1, 2]), (k, v) -> k || CAST(v as VARCHAR)); -- {a1=1, b2=2}

SELECT transform_keys(MAP(ARRAY [1, 2], ARRAY [1.0, 1.4]), (k, v) -> MAP(ARRAY[1, 2], ARRAY['one', 'two'])[k]); -- {two=1.4, one=1.0}
```

- size(*x*) → bigint

  Description: Returns the capacity of Map(x) **x**.

```
select size(map(array['num1','num2'],array[11,12])); --2
```

- transform_values(*map(K, V1)*, *function(K*, *V2*, *V2)) -> map(K*, *V2*)

  Description: Maps value **V1** to value **V2** for each entry in the map and keeps the corresponding key unchanged.

```
SELECT transform_values(MAP(ARRAY[], ARRAY[]), (k, v) -> v + 1); -- {}

SELECT transform_values(MAP(ARRAY [1, 2, 3], ARRAY [10, 20, 30]), (k, v) -> v + k); -- {1=11, 2=22, 3=33}

SELECT transform_values(MAP(ARRAY [1, 2, 3], ARRAY ['a', 'b', 'c']), (k, v) -> k * k); -- {1=1, 2=4, 3=9}

SELECT transform_values(MAP(ARRAY ['a', 'b'], ARRAY [1, 2]), (k, v) -> k || CAST(v as VARCHAR)); -- {a=a1, b=b2}

SELECT transform_values(MAP(ARRAY [1, 2], ARRAY [1.0, 1.4]),(k, v) -> MAP(ARRAY[1, 2], ARRAY['one', 'two'])[k] || '_' || CAST(v AS VARCHAR)); -- {1=one_1.0, 2=two_1.4}
```

11.13.6.18 URL Function

Extraction Function

Description: Extracts content from an HTTP URL (or any URL that complies with the RFC 2396 standard).

```
[protocol:][//host[:port]][path][?query][#fragment]
```

The extracted content does not contain URI syntax separators, such as **:** or **?**.

- url_extract_fragment(*url*) → varchar

  Description: Returns the segment identifier of the URL, that is, the character string following #.

  ```
  select url_extract_fragment('http://www.example.com:80/stu/index.html?
  name=xxx&age=25#teacher');--teacher
  ```

- url_extract_host(url) → varchar

  Description: Returns the host domain name in *url*.

  ```
  select url_extract_host('http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher');--
  www.example.com
  ```

- url_extract_parameter(*url*, *name*) → varchar

  Description: Returns the **name** parameter in *url*.

  ```
  select url_extract_parameter('http://www.example.com:80/stu/index.html?
  name=xxx&age=25#teacher','age');-- 25
  ```

- url_extract_path(*url*) → varchar

  Description: Extracts the path from *url*.

  ```
  select url_extract_path('http://www.example.com:80/stu/index.html?
  name=xxx&age=25#teacher');-- /stu/index.html
  ```

- url_extract_port(*url*) → bigint

  Description: Extracts the port number from *url*.

  ```
  select url_extract_port('http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher');-- 80
  ```

- url_extract_protocol(*url*) → varchar

  Description: Extracts the protocol from *url*.

  ```
  select url_extract_protocol('http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher');
  -- http
  ```

- url_extract_query(*url*) → varchar

  Description: Extracts the query character string from *url*.

  ```
  select url_extract_query('http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher'); --
  name=xxx&age=25
  ```

Encoding Function

- url_encode(*value*) → varchar

  Description: Escapes *value* so that it can be securely contained in the URL query parameter name and value.

  - Letter characters are not encoded.
  - Characters **.**, **-**, *****, and _ are not encoded.
  - ASCII space characters are encoded as **+**.
  - All other characters are converted to UTF-8, and the byte is encoded as a string **%***XX*, where *XX* is an uppercase hexadecimal value of UTF-8 bytes.

  ```
  select url_encode('http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher');
  -- http%3A%2F%2Fwww.example.com%3A80%2Fstu%2Findex.html%3Fname%3Dxxx%26age
  %3D25%23teacher
  ```

- url_decode(*value*) → varchar

  Description: Decodes the URL after value encoding.

  ```
  select url_decode('http%3A%2F%2Fwww.example.com%3A80%2Fstu%2Findex.html%3Fname%3Dxxx
  %26age%3D25%23teacher');
  -- http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher
  ```


11.13.6.19 Geospatial Function

The HetuEngine Geospatial function starting with **ST_** supports SQL and MM specifications and complies with the Open GIS specifications of Open Geospatial Consortium (OGC). Therefore, many HetuEngine Geospatial features require, or more accurately put, assume that the geometry to be manipulated is both simple and efficient. For example, it makes no sense to calculate the area of a polygon that defines holes outside the polygon, or to construct a polygon from non-simple boundary lines.

The HetuEngine geospatial function supports both known text (WKT) and known binary (WKB) forms of spatial objects:

- POINT (0 0)
- LINESTRING (0 0, 1 1, 1 2)
- POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1))
- MULTIPOINT (0 0, 1 2)
- MULTILINESTRING ((0 0, 1 1, 1 2), (2 3, 3 2, 5 4))
- MULTIPOLYGON (((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1)), ((-1 -1, -1 -2, -2 -2, -2 -1, -1 -1)))
- GEOMETRYCOLLECTION (POINT(2 3), LINESTRING (2 3, 3 4))

Well-known Text (WKT) is a text markup language developed by the Open Geospatial Consortium (OGC (Open GIS Consortium)). It is used to represent vector geometric objects, spatial reference systems, and conversion between spatial reference systems.

Well-known Binary (WKB) is a binary representation of WKT. It solves the problem of redundant WKT expressions and facilitates the transmission and storage of the same information in the database.

GeoJSON is a feature information output format in JSON format. It can be easily processed by script languages such as JavaScript. Geographic databases such as OpenLayers use the GeoJSON format. In addition, more simplified extended formats such as TopoJSON are used.

Use the **ST_GeometryFromText()** and **ST_GeomFromBinary()** functions to create geometric objects from WKT or WKB.

The SphericalGeography type provides local support for spatial elements represented on geographic coordinates (sometimes referred to as geodetic coordinates or **lat / lon** or **lon / lat)**. Geographical coordinates are spherical coordinates expressed in degrees. Geometric types are based on the plane. The shortest path between two points on the plane is a straight line. This means that Cartesian mathematics and linear vectors can be used to calculate geometric shapes (area, distance, length, intersection, etc.).

The SphericalGeography type is based on a sphere. The shortest path between two points on the sphere is the large arc. This means that more complex mathematical methods must be used to calculate the terrain (areas, distances, lengths, intersections, etc.) on the sphere. More accurate measurements that take into account the actual shape of the sphere are not supported.

Values returned by the measurement functions **ST_Distance ()** and **ST_Length ()** are in meters. **ST_Area()** returns a value in square meters.

Use the **to_spherical_geography ()** function to convert a geometric object to a geographic object.

For example, **ST_Distance(ST_Point(-71.0882, 42.3607)**, **ST_Point(-74.1197, 40.6976))** return **3.4577** in units of input values on the Euclidean plane. **ST_Distance (to_spherical_geography (ST_Point(-71.0882, 42.3607)), to_spherical_geography (ST_Point(-74.1197, 40.6976)))** returns the value **312822.179** in meters.

Constructor

- ST_AsBinary(*Geometry*) → varbinary

  Description: Returns the binary representation of a geometry.

  ```
  select ST_AsBinary(ST_GeometryFromText('POINT(12 13)'));
                      _col0
  ------------------------------------------------------------------
  01 01 00 00 00 00 00 00 00 00 00 28 40 00 00 00 00 00 00 2a 40
  (1 row)
  ```

- ST_AsText(*Geometry*) → varchar

  Description: Returns the WKT representation of a geometry. For an empty geometric figure, **ST_AsText(ST_LineFromText('LINESTRING EMPTY'))** generates **'MULTI LINE STRING EMPTY'**, and **ST_AsText(ST_Polygon('POLYGON EMPTY'))** generates **'MULTIPOLYGON EMPTY'**.

  ```
  SELECT st_astext(ST_GeometryFromText('LINESTRING (0 0, 1 1, 1 2)'));-- LINESTRING (0 0, 1 1, 1 2)
  ```

- ST_GeometryFromText(*varchar*) → Geometry

  Description: Returns a geometric object represented by WKT.

  ```
  select ST_GeometryFromText('POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1))');
  --POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 1 2, 2 2, 2 1, 1 1))
  ```

- ST_GeomFromBinary(*varbinary*) → Geometry

  Description: Returns a geometric type represented by WKB.

  ```
  select ST_geomFromBinary(ST_AsBinary(ST_GeometryFromText('POINT(12 13)')));-- POINT (12 13)
  ```

- ST_LineFromText(*varchar*) → LineString

  Description: Returns the line string object represented by WKT.

  ```
  select st_lineFromText('LINESTRING (0 0, 1 1, 1 2)');--  LINESTRING (0 0, 1 1, 1 2)
  ```

- ST_Point(*double*, *double*)

  Description: Returns a geometric point object with a given coordinate value.

  ```
  select st_point(12.1,34.1);
  POINT (12.1 34.1)
  ```

- ST_Polygon(*varchar*)

  Description: Returns the polygon represented by the WKT string.

  ```
  select ST_Polygon('POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1))');
  POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 1 2, 2 2, 2 1, 1 1))
  ```

Operations

- ST_Boundary(*Geometry*) → Geometry

  Description: Returns the boundary of a closed graph.

  ```
  select ST_boundary(ST_Polygon('POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1))'));
  MULTILINESTRING ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 1 2, 2 2, 2 1, 1 1))
  ```

- ST_Buffer(*Geometry*, *distance*) → Geometry

Description: Returns the geometry representing all points whose distance to the specified geometry is less than or equal to the specified distance.

```
select ST_Buffer(ST_POINT(0,0),4);
POLYGON ((4 0, 3.9914356929544113 0.2616125169205717, 3.965779445495239
0.5221047688802056, 3.923141121612919 0.780361288064512 2, 3.8637033051562706
1.035276180410082, 3.7877205179804205 1.2857578612126452, 3.695518130045145
1.5307337294603...
```

- ST_Difference(*Geometry*, *Geometry*) → Geometry

  Description: Returns the geometry value representing the point set difference for a given geometry.

  ```
  select ST_Difference(ST_POINT(0,0),ST_POINT(2,3));-- POINT (0 0)
  ```

- ST_EnvelopeAsPts(*Geometry) -> array(Geometry*)

  Description: Returns an array of two points: the lower left and upper right corners of a bounding rectangle polygon of a geometry. If the input geometry is empty, **null** is returned.

  ```
  select ST_EnvelopeAsPts(ST_LineFromText('LINESTRING (0 0, 1 1, 1 2)'));--  [POINT (0 0), POINT (1 2)]
  ```

- ST_Intersection(*Geometry*, *Geometry*) → Geometry

  Description: Returns the geometric value representing the intersection of two geometric point sets.

  ```
  select ST_Intersection(ST_LineFromText('LINESTRING (0 0, 1 1, 1 2)'), ST_LineFromText('LINESTRING
  (0 0, 1 1, 1 3)'));
  -- LINESTRING (0 0, 1 1, 1 2)
  ```

- ST_SymDifference(*Geometry*, *Geometry*) → Geometry

  Description: Returns a geometric value representing the symmetric difference between the point sets of two geometries.

  ```
  select ST_SymDifference(ST_LineFromText('LINESTRING (0 0, 1 1, 1 2)'),
  ST_LineFromText('LINESTRING (0 0, 1 1, 1 3)'));--  LINESTRING (1 2, 1 3)
  ```

- ST_Union(*Geometry*, *Geometry*) → Geometry

  Description: Returns a geometry representing the union of the points of the input geometry.

  ```
  select ST_Union(ST_LineFromText('LINESTRING (0 0, 1 1, 1 2)'), ST_LineFromText('LINESTRING (0 0, 1
  1, 1 3)')); --  LINESTRING (0 0, 1 1, 1 2, 1 3)
  ```

11.13.6.20 HyperLogLog Functions

HetuEngine uses the HyperLogLog data structure to implement the **rox_distinct ()** function.

Data Structure

HyperLogLog (hll) is a statistical base algorithm. It does not store the number of occurrences of each element. It uses the probability algorithm to calculate the number of elements by storing the location of the first 1 in the 32-bit hash value of the element. Generally, there are two types of storage structures: a sparse storage structure and a dense storage structure. HLL is created in a sparse storage structure. When more efficient processing is required, HLL is converted to intensive data structures. P4HyperLogLog is a dense data structure in its rectification life cycle. If necessary, run **cast(hll as P4HyperLogLog)** to convert it explicitly. In the current implementation of the data engine, the HLL data sketch uses a group of 32-bit buckets to store the maximum hash value.

Serialization

Data sketches can be serialized and deserialized using varbinary. This allows them to be easily stored for later use. By combining multiple sketches, we can query **approx_distinct()** of all elements in the partition, that is, the approximate number of occurrences of each element, and then complete the entire query with a small overhead.

For example, if you only need to calculate the number of times that each user browses web pages every day, you can calculate the weekly and yearly data by accumulating the data. This is similar to calculating the weekly revenue by summarizing the daily revenue.

You can use approx_distinct() together with GROUPING SETS to convert it to HyperLogLog. The following is an example:

```
CREATE TABLE visit_summaries(visit_date date,hll varbinary);

INSERT INTO visit_summaries
SELECT visit_date,cast(approx_set(user_id) AS varbinary)
FROM user_visits
GROUP BY visit_date;

SELECT cardinality(merge(cast(hll AS HyperLogLog)))AS weekly_unique_users
FROM visit_summaries
WHERE visit_date>=current_date-interval'7'day;
```

Function

- approx_set(*x*) → HyperLogLog

  Description: Returns HyperLogLog. This data sketch is the basis of **approx distinct()** and can be stored and used by calling **cardinality()**.

  ```
  select approx_set(cookieid) from cookies_log;--02 0c 02 00 c0 77 15 40 c1 2f 1b c2
  ```

- cardinality(*hll*) → bigint

  Description: Calculates the data summarized by HLL.

  ```
  select cardinality(approx_set(cookieid)) from cookies_log; --2
  ```

- empty_approx_set() → HyperLogLog

  Description: Returns an empty HyperLogLog.

  ```
  select empty_approx_set();--02 0c 00 00
  ```

- merge(*HyperLogLog*) → HyperLogLog

  Description: Summarizes the union set of each independent HLL data sketch.

  ```
  CREATE TABLE visit_summaries (  visit_date date,  hll varbinary);

  insert into visit_summaries select createtime,cast(approx_set(cookieid) as varbinary) from cookies_log
  group by createtime;

  SELECT cardinality(merge(cast(hll AS HyperLogLog))) AS weekly_unique_users FROM visit_summaries
  WHERE visit_date >=date '2020-07-11';
   weekly_unique_users
  ---------------------
                     2
  (1 row)
  ```

11.13.6.21 UUID Function

- Syntax

  uuid()

- Description

  This function is used to generate a pseudo-random unique universal identifier.

- Example

```
select uuid();
```

11.13.6.22 Color Function

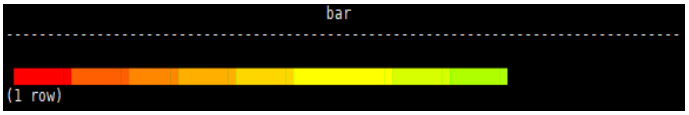
- bar(*x*, *width*)

  Description: Renders a single bar in the ANSI bar chart using the default low-frequency red and high-frequency green. For example, if you pass 25% of *x* and 40 widths to this function. A 10-character red bar followed by 30 spaces will be drawn to create a 40-character bar.

- bar(x, width, low_color, high_color)

  Description: Draws a straight line with the specified width in an ANSI bar chart. The x parameter is a double-precision value ranging from 0 to 1. If the value of *x* is out of the range [0,1], the value is truncated to 0 or 1. **low_color** and **high_color** capture the color at either end of the horizontal bar chart. For example, if *x* is **0.5**, *width* is **80**, **low_color** is **0xFF0000**, and **high_color** is **0x00FF00**, this function returns a 40-character bar. The bar consists of red (0xFF0000) and yellow (0xFFFF00), the remaining 80 characters are padded with spaces.

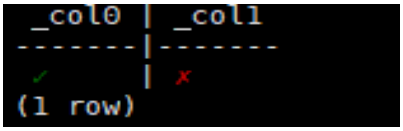
```
select bar(0.75,80,rgb(255,0,0),rgb(0,255,0));
```



- render(b)

  Description: Returns right and wrong symbols based on Boolean values.

```
select render(true),render(false);
```



11.13.6.23 Session Information

current_user

Description: Returns the current user.

```
select current_user;
```

current_user()

Description: Returns the current user.

current_catalog

Description: Returns the current catalog name.

```
select current_catalog;
```

current_schema

Description: Returns the current schema name.


```
select current_schema;
```

11.13.6.24 Teradata Function

The following functions provide the Teradata SQL capability.

String Functions

- char2hexint(*string*)

  Description: Returns the hexadecimal representation of the UTF-16BE encoding of a string.

- index(*string*, *substring*)

  Description: Same as strpos().

Date Functions

The functions in this section use format strings that are compatible with the Teradata datetime function. The following table describes the supported format specifiers based on the Teradata reference manual:

Specifier	Description
- / , . ; :	Ignore punctuation
dd	Day (1 to 31) in a month
hh	Hour (1 to 12) in a day
hh24	Hour (0 to 23) in a day
mi	Minute (0 to 59)
mm	Month (01 to 12)
ss	Second (0 to 59)
yyyy	Four-digit year
yy	Two-digit year

NOTE

Case-insensitive is not supported. All specifiers must be in lower case.

- to_char(*timestamp*, *format*)

  Description: Outputs a timestamp as a string in a specified format.

  ```
  select to_char(timestamp '2020-12-18 15:20:05','yyyy/mmdd hh24:mi:ss');-- 2020/1218 15:20:05
  ```

- to_timestamp(*string*, *format*)

  Description: Parses a string to a timestamp in a specified format.

  ```
  select to_timestamp('2020-12-18 15:20:05','yyyy-mm-dd hh24:mi:ss'); -- 2020-12-18 15:20:05.000
  ```

- to_date(*string*, *format*)

  Description: Converts a string to a date in the specified format.


```
select to_date('2020/12/04','yyyy/mm/dd'); -- 2020-12-04
```

11.13.6.25 Data Masking Functions

Data masking refers to the process of distorting sensitive information based on masking rules to protect sensitive privacy data.

- mask_first_n(string str[, int n]) →varchar

  Description: Returns the masked version of a string. The first *n* values are masked. Uppercase letters are converted to **X**, lowercase letters to **x**, and digits to **n**.

  ```
  select mask_first_n('Aa12-5678-8765-4321', 4);
        _col0
  ---------------------
   Xxnn-5678-8765-4321
  (1 row)
  ```

- mask_last_n(string str[, int n]) →varchar

  Description: Returns the masked version of a string. The last *n* values are masked. Uppercase letters are converted to **X**, lowercase letters to **x**, and digits to **n**.

  ```
  select mask_last_n('1234-5678-8765-Hh21', 4);
        _col0
  ---------------------
   1234-5678-8765-Xxnn
  (1 row)
  ```

- mask_show_first_n(string str[, int n]) →varchar

  Description: Returns the masked version of a string. Only the first *n* characters are displayed. Uppercase letters are converted to **X**, lowercase letters to **x**, and digits to **n**.

  ```
  select mask_show_first_n('1234-5678-8765-4321',4);
        _col0
  ---------------------
   1234-nnnn-nnnn-nnnn
  (1 row)
  ```

- mask_show_flairst_n(string str[, int n]) →varchar

  Description: Returns the masked version of a string. Only the last *n* characters are displayed. Uppercase letters are converted to **X**, lowercase letters to **x**, and digits to **n**.

  ```
  select mask_show_last_n('1234-5678-8765-4321',4);
        _col0
  ---------------------
   nnnn-nnnn-nnnn-4321
  (1 row))
  ```

- mask_hash(string|char|varchar str) →varchar

  Description: Returns a hash value of a string. Hash values are consistent and can be used to join masked values across tables. For a non-string, **NULL** is returned.

  ```
  select mask_hash('panda');
                              _col0
  ----------------------------------------------------------------------
   a7cdf5d0586b392473dd0cd08c9ba833240006a8a7310bf9bc8bf1aefdfaeadb
  (1 row)
  ```

11.13.6.26 IP Address Functions

contains(network, address) → boolean

If the CIDR network contains the **address** value, **true** is returned.

Example:

- true
```
SELECT contains('10.0.0.0/8', IPADDRESS '10.255.255.255');

SELECT contains('2001:0db8:0:0:0:ff00:0042:8329/128', IPADDRESS '2001:0db8:0:0:0:ff00:0042:8329');
```
- false
```
SELECT contains('10.0.0.0/8', IPADDRESS '11.255.255.255');

SELECT contains('2001:0db8:0:0:0:ff00:0042:8329/128', IPADDRESS '2001:0db8:0:0:0:ff00:0042:8328');
```

11.13.6.27 Quantile Digest Functions

Overview

Quantile digest is a data sketch that stores approximate percentile information. The HetuEngine type for this data structure is called qdigest.

Function

- merge(qdigest) → qdigest

  Description: Merges all input qdigest data into one qdigest.
- value_at_quantile(qdigest(T), quantile) → T

  Description: Returns the approximate percentile values from the quantile digest given the number **quantile** between 0 and 1.
- values_at_quantiles(qdigest(T), quantiles) -> array(T)

  Description: Returns the approximate percentile values as an array given the input quantile digest and array of values between 0 and 1 which represent the quantiles to return.
- qdigest_agg(x) -> qdigest([same as x])

  Description: Returns the **qdigest** which is composed of all input values of x.
- qdigest_agg(x, w) -> qdigest([same as x])

  Description: Returns the **qdigest** which is composed of all input values of **x** using the per-item weight **w**.
- qdigest_agg(x, w, accuracy) -> qdigest([same as x])

  Description: Returns the **qdigest** which is composed of all input values of **x** using the per-item weight **w** and maximum error of accuracy. **accuracy** must be a value greater than zero and less than one, and it must be constant for all input rows.

11.13.6.28 T-Digest Functions

Overview

A T-digest is a data sketch that stores approximate percentile information. The HetuEngine type for this data structure is called tdigest. Tdigests may be merged without losing precision, and for storage and retrieval they may be cast to/from VARBINARY.

Function

- merge(tdigest) → tdigest

  Description: Merges all input tdigests into a single tdigest.

- value_at_quantile(tdigest,quantile) → double

  Description: Returns the approximate percentile values from the T-digest given the number **quantile** between 0 and 1.

- values_at_quantiles(tdigest,quantiles)->array(double)

  Description: Returns the approximate percentile values as an array given the input T-digest and array of values between 0 and 1 which represent the quantiles to return.

- tdigest_agg(x)->tdigest

  Description: Returns the **tdigest** which is composed of all input values of x. **x** can be of any numeric type.

- tdigest_agg(x,w)->tdigest

  Description: Returns the **tdigest** which is composed of all input values of **x** using the per-item weight **w**. **w** must be no less than 1. **x** and **w** can be of any numeric type.

11.13.6.29 Set Digest Functions

Overview

HetuEngine provides several functions that use the MinHash technique.

MinHash quickly estimates how similar two sets based on their Jaccard similarity coefficients. It is usually used in data mining to detect almost the same web pages at scale. By using this information, search engines effectively avoid displaying two web pages that are nearly identical in search results.

The following example shows how to use set digest functions to estimate the similarity between texts. The input texts are split into 4-shingles by the **ngrams()** function. (The texts are divided into consecutive subsequences each is 4 characters long, and each subsequence is called a shingle or gram.) They are used to create a set digest of each initial text. The set digests are compared with each other to obtain an approximation of similarity of their initial texts.

```
WITH text_input(id, text) AS (
        VALUES
            (1, 'The quick brown fox jumps over the lazy dog'),
            (2, 'The quick and the lazy'),
            (3, 'The quick brown fox jumps over the dog')
    ),
   text_ngrams(id, ngrams) AS (
       SELECT id,
              transform(
                ngrams(
                  split(text, ' '),
                  4
                ),
                token -> array_join(token, ' ')
              )
        FROM text_input
    ),
    minhash_digest(id, digest) AS (
        SELECT id,
```



```
            (SELECT make_set_digest(v) FROM unnest(ngrams) u(v))
        FROM text_ngrams
    ),
    setdigest_side_by_side(id1, digest1, id2, digest2) AS (
        SELECT m1.id as id1,
               m1.digest as digest1,
               m2.id as id2,
               m2.digest as digest2
        FROM (SELECT id, digest FROM minhash_digest) m1
        JOIN (SELECT id, digest FROM minhash_digest) m2
          ON m1.id != m2.id AND m1.id < m2.id
    )
SELECT id1,
       id2,
       intersection_cardinality(digest1, digest2) AS intersection_cardinality,
       jaccard_index(digest1, digest2)
           AS jaccard_index
FROM setdigest_side_by_side
ORDER BY id1, id2;
 id1 | id2 | intersection_cardinality | jaccard_index
-----|-----|--------------------------|---------------
   1 |   2 |                        0 |           0.0
   1 |   3 |                        4 |           0.6
   2 |   3 |                        0 |           0.0
(3 rows)
```

The above list indicates that, as expected, the texts with IDs **1** and **3** are similar.

NOTE

Data sketches can be serialized to varbinary or deserialized from varbinary. Varbinary can be used to store data sketches.

Function

- make_set_digest(x) → setdigest

  Description: Composes all input values of **x** into a **setdigest**.

  ```
  SELECT make_set_digest(value) FROM (VALUES 1, 2, 3) T(value);
                       _col0
  ----------------------------------------------------
   01 10 00 00 00 02 0b 03 00 80 03 44 00 00 58 3d
   5b 80 20 08 de 00 20 00 00 03 00 00 00 a8 c0 76
   6c a0 20 08 de 4a c4 05 fb b7 03 44 00 0c 8b 48
   b2 39 58 3d 5b 01 00 01 00 01 00
  (1 row)

  SELECT make_set_digest(value) FROM (VALUES 'Trino', 'SQL', 'on', 'everything') T(value);
                       _col0
  ----------------------------------------------------
   01 14 00 00 00 02 0b 04 00 c0 8c 7d 1e c0 75 c9
   2d c0 1a 1a 66 03 11 c3 a5 00 20 00 00 04 00 00
   00 06 e5 2d 45 05 11 c3 a5 48 85 6b d5 e0 8c 7d
   1e b9 1a 8a 39 ff 75 c9 2d 02 ad 0c 7c ed 1a 1a
   66 01 00 01 00 01 00 01 00
  (1 row)
  ```

- merge_set_digest(setdigest) → setdigest

  Description: Returns the **setdigest** of the aggregate union of the individual **setdigest** structures.

- cardinality(setdigest) → long

  Description: Returns the cardinality of the **setdigest** from its internal HyperLogLog component.

  ```
  SELECT cardinality(make_set_digest(value)) FROM (VALUES 1, 2, 2, 3, 3,4, 4, 4, 5) T(value); -- 5
  ```

- intersection_cardinality(x, y) → long

  Description: Returns the estimation for the cardinality of the intersection of the two set digests. **x** and **y** must be of the **setdigest** type.

  ```
  SELECT intersection_cardinality(make_set_digest(v1), make_set_digest(v2)) FROM (VALUES (1, 1), (NULL, 2), (2, 3), (3, 4)) T(v1, v2); -- 3
  ```

- jaccard_index(x, y) → double

  Description: Returns the estimation of Jaccard index for the two set digests.aries. **x** and **y** must be of the **setdigest** type.

  ```
  SELECT jaccard_index(make_set_digest(v1), make_set_digest(v2)) FROM (VALUES (1, 1), (NULL,2), (2, 3), (NULL, 4)) T(v1, v2); -- 0.5
  ```

- hash_counts(x)

  Description: Returns a map containing the Murmur3Hash128 hashed values and the count of their occurrences within the internal **MinHash** structure belonging to **x**. **x** must be of the **setdigest** type.

  ```
  SELECT hash_counts(make_set_digest(value)) FROM (VALUES 1, 1, 1, 2, 2) T(value); -- {19144387141682250=3, -2447670524089286488=2}
  ```

11.13.7 HetuEngine Auxiliary Command Syntax

11.13.7.1 USE

Syntax

- USE catalog.schema
- USE schema

Description

This statement is used to specify the catalog and schema used by the current session. If the catalog is not specified, the current catalog is used by default.

Example

To specify the current session to the schema named **test** in the Hive catalog:

```
USER hive.test
```

Precautions

None

11.13.7.2 SET SESSION

- Syntax

  SET SESSION name = expression;

  SET SESSION catalog.name = expression;

- Description

  This statement is used to set the specified properties of the current session.

- Example

  ```
  SET SESSION optimize_hash_generation = true;
  SET SESSION hive.optimized_reader_enabled = true;
  ```


11.13.7.3 RESET SESSION

Syntax

- RESET SESSION name
- RESET SESSION catalog.name

Description

This statement is used to reset the specified properties of the current session.

Example

```
RESET SESSION optimize_hash_generation;
RESET SESSION hive.optimized_reader_enabled;
```

11.13.7.4 DESCRIBE

Syntax

DESCRIBE [EXTENDED| FORMATTED] table_name

DESCRIBE [EXTENDED| FORMATTED] table_name PARTITION (partition_spec)

Description

This statement is used to view the metadata information of a specified table. Currently, this syntax can display only the metadata of columns, which is equivalent to the SHOW COLUMNS syntax.

After the **EXTENDED** keyword is added, all metadata of the table is displayed in the Thrift serialization format.

If the **FORMATTED** keyword is added, the metadata of the table is displayed in a table.

Example

Display the column information of the **fruit** data table:

```
DESCRIBE fruit;
```

Display the Fruit metadata:

```
DESCRIBE FORMATTED fruit;
                      Describe Formatted Table
-------------------------------------------------------------------------------
# col_name      data_type      comment
name      varchar
price      integer

# Detailed Table Information
Database:                default
Owner:                   admintest
LastAccessTime:          0
Location:                hdfs://hacluster/user/hive/warehouse/fruit
Table Type:              MANAGED_TABLE

# Table Parameters:
```



```
    Owner                   ggg
    STATS_GENERATED_VIA_STATS_TASK  workaround for potential lack of HIVE-12730
    numFiles                0
    numRows                 0
    orc.compress.size       262144
    orc.compression.codec   GZIP
    orc.row.index.stride    10000
    orc.stripe.size         67108864
    presto_query_id         20210308_072339_00075_5ck2k@default@HetuEngine
    presto_version
    rawDataSize             0
    totalSize               0
    transient_lastDdlTime   1615188219

# Storage Information
SerDe Library:              org.apache.hadoop.hive.ql.io.orc.OrcSerde
InputFormat:                org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
OutputFormat:               org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
Compressed:                 No
Num Buckets:                -1
Bucket Columns:             []
Sort Columns:               []
serialization.format:       1
(1 row)

Query 20210309_022835_00007_2i9yy@default@HetuEngine, FINISHED, 1 node
Splits: 1 total, 1 done (100.00%)
0:01 [0 rows, 0B] [0 rows/s, 0B/s];
```

11.13.7.5 DESCRIBE FORMATTED COLUMNS

Syntax

DESCRIBE FORMATTED [db_name.]table_name [PARTITION partition_spec] col_name

Description

This statement is used to describe the column information of a table or partition.

Collect statistics on the columns of specified tables or partitions.

The following is an example.

Example

```
describe formatted show_table1 a;
  Describe Formatted Column
-------------------------------
 col_name             a
 data_type            integer
 min
 max
 num_nulls
 distinct_count       0
 avg_col_len
 max_col_len
 num_trues
 num_falses
 comment
(1 row)
```


11.13.7.6 DESCRIBE DATABASE| SCHEMA

Syntax

DESCRIBE DATABASE|SCHEMA [EXTENDED] schema_name

Description

DATABASE and **SCHEMA** are equivalent and interchangeable. They have the same meaning.

This statement is used to display the name, comment, and root path of a schema on the file system.

The option **EXTENDED** can be used to display the database attributes of the schema.

Example

```
CREATE SCHEMA web;

DESCRIBE SCHEMA web;
                        Describe Schema
------------------------------------------------------------------------------
 web      hdfs://hacluster/user/hive/warehouse/web.db   admintest   USER
(1 row)
```

11.13.7.7 DESCRIBE INPUT

Syntax

DESCRIBE INPUT statement_name

Description

This statement is used to list the input parameters of the prepared statement, parameter positions, and the type of each input parameter. **Unknown** is displayed if the parameter type is not determined.

Example

- The following statement is used to prepare a precompiled statement with three input parameters and list the parameters of the precompiled statement.
  ```
  PREPARE my_select1 FROM SELECT ? FROM fruit WHERE name  = ? AND price < ?;
  DESCRIBE INPUT my_select1;
  ```
- A precompiled statement without input parameters:
  ```
  PREPARE my_select2 FROM SELECT * FROM fruit;
  DESCRIBE INPUT my_select2;
  ```

11.13.7.8 DESCRIBE OUTPUT

Syntax

DESCRIBE OUTPUT statement_name

Description

This statement is used to list the output columns of the prepared statement.

Including the column name (or alias), catalog, schema, table name, type, type size (in bytes).

And a boolean value indicating whether the column is an alias.

Example

```
--PREPARE my_select1 FROM SELECT * FROM fruit;
  DESCRIBE OUTPUT my_select1;
--PREPARE my_select2 FROM SELECT count(*) as my_count, 1+2 FROM fruit;
  DESCRIBE OUTPUT my_select2;
--PREPARE my_create FROM CREATE TABLE foo AS SELECT * FROM fruit;
  DESCRIBE OUTPUT my_create;
```

11.13.7.9 EXPLAIN

Syntax

EXPLAIN [(option [, ...])] statement

The option can be the following:

FORMAT { TEXT | GRAPHVIZ | JSON }

TYPE { LOGICAL | DISTRIBUTED | VALIDATE | IO}

Description

This statement is used to display the logical or distributed execution plan of a statement. It can also be used to verify an SQL statement or analyze I/Os.

The TYPE DISTRIBUTED parameter is used to display the fragmented plan. Each fragment is executed by one or more nodes. **Fragments separation** indicates that data is exchanged between two nodes. **Fragment type** indicates how a fragment is executed and how data is distributed among different fragments.

- SINGLE

  Fragments are executed on a single node.
- HASH

  Fragments are executed on a fixed number of nodes, and the input data is distributed using the hash function.
- ROUND_ROBIN

  Fragments are executed on a fixed number of nodes, and input data is distributed in round-robin mode.
- BROADCAST

  Fragments are executed on a fixed number of nodes, and the input data is broadcast to all nodes.
- SOURCE

  Fragments are executed on the node that accesses the input fragments.

Example

- LOGICAL:

```
CREATE TABLE testTable (regionkey int, name varchar);
EXPLAIN SELECT regionkey, count(*) FROM testTable GROUP BY 1;
                                                  Query Plan
------------------------------------------------------------------------------------------------------------------------
--------------
 Output[regionkey, _col1]
 │   Layout: [regionkey:integer,
count:bigint]
 │   Estimates: {rows: ? (?), cpu: ?, memory: ?,
network: ?}
 │   _col1 := count
 └─
RemoteExchange[GATHER]
    │   Layout: [regionkey:integer,
count:bigint]
    │   Estimates: {rows: ? (?), cpu: ?, memory: ?,
network: ?}
    └─ Project[]
        │   Layout: [regionkey:integer,
count:bigint]
        │   Estimates: {rows: ? (?), cpu: ?, memory: ?,
network: ?}
        └─ Aggregate(FINAL)[regionkey]
[$hashvalue]
            │   Layout: [regionkey:integer, $hashvalue:bigint,
count:bigint]
            │   Estimates: {rows: ? (?), cpu: ?, memory: ?,
network: ?}
            │   count := count("count_8")
            └─ LocalExchange[HASH][$hashvalue]
("regionkey")
                │   Layout: [regionkey:integer, count_8:bigint,
$hashvalue:bigint]
                │   Estimates: {rows: ? (?), cpu: ?, memory: ?,
network: ?}
                └─ RemoteExchange[REPARTITION]
[$hashvalue_9]
                    │   Layout: [regionkey:integer, count_8:bigint,
$hashvalue_9:bigint]
                    │   Estimates: {rows: ? (?), cpu: ?, memory: ?,
network: ?}
                    └─ Aggregate(PARTIAL)[regionkey]
[$hashvalue_10]
                        │   Layout: [regionkey:integer, $hashvalue_10:bigint,
count_8:bigint]
                        │   count_8 := count(*)
                        └─ ScanProject[table =
hive:default:testtable]
                                Layout: [regionkey:integer,
$hashvalue_10:bigint]
                                Estimates: {rows: 0 (0B), cpu: 0, memory: 0B, network: 0B}/{rows: 0 (0B), cpu: 0,
memory: 0B, network: 0B}
                                $hashvalue_10 := "combine_hash"(bigint '0', COALESCE("$operator
$hash_code"("regionkey"), 0))
                                regionkey := regionkey:int:0:REGULAR
```

- DISTRIBUTED:

```
EXPLAIN (type DISTRIBUTED) SELECT regionkey, count(*) FROM testTable GROUP BY 1;
                                              Query Plan
------------------------------------------------------------------------------------------------------------------------
 Fragment 0 [SINGLE]
     Output layout: [regionkey, count]
     Output partitioning: SINGLE []
     Stage Execution Strategy:
UNGROUPED_EXECUTION
     Output[regionkey, _col1]
     │   Layout: [regionkey:integer, count:bigint]
```

```
    │  Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
    │  _col1 := count
    └─ RemoteSource[1]
         Layout: [regionkey:integer, count:bigint]

 Fragment 1 [HASH]
    Output layout: [regionkey, count]
    Output partitioning: SINGLE []
    Stage Execution Strategy:
UNGROUPED_EXECUTION
    Project[]
    │  Layout: [regionkey:integer, count:bigint]
    │  Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
    └─ Aggregate(FINAL)[regionkey][$hashvalue]
       │  Layout: [regionkey:integer, $hashvalue:bigint, count:bigint]
       │  Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
       │  count := count("count_8")
       └─ LocalExchange[HASH][$hashvalue] ("regionkey")
          │  Layout: [regionkey:integer, count_8:bigint, $hashvalue:bigint]
          │  Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
          └─ RemoteSource[2]
               Layout: [regionkey:integer, count_8:bigint, $hashvalue_9:bigint]

 Fragment 2 [SOURCE]
    Output layout: [regionkey, count_8, $hashvalue_10]
    Output partitioning: HASH [regionkey][$hashvalue_10]
    Stage Execution Strategy:
UNGROUPED_EXECUTION
    Aggregate(PARTIAL)[regionkey][$hashvalue_10]
    │  Layout: [regionkey:integer, $hashvalue_10:bigint, count_8:bigint]
    │  count_8 := count(*)
    └─ ScanProject[table = hive:default:testtable, grouped = false]
         Layout: [regionkey:integer, $hashvalue_10:bigint]
         Estimates: {rows: 0 (0B), cpu: 0, memory: 0B, network: 0B}/{rows: 0 (0B), cpu: 0, memory: 0B,
network: 0B}
         $hashvalue_10 := "combine_hash"(bigint '0', COALESCE("$operator$hash_code"("regionkey"),
0))
         regionkey := regionkey:int:0:REGULAR
```

- VALIDATE:

```
EXPLAIN (TYPE VALIDATE) SELECT id, count(*) FROM testTable GROUP BY 1;
 Valid
-------
 true
```

- I/O:

```
EXPLAIN (TYPE IO, FORMAT JSON) SELECT regionkey , count(*) FROM testTable GROUP BY 1;

          Query Plan
---------------------------------
{
  "inputTableColumnInfos" : [ {
    "table" : {
      "catalog" : "hive",
      "schemaTable" : {
        "schema" : "default",
        "table" : "testtable"
      }
    },
    "columnConstraints" : [ ]
  } ]
}
```

11.13.7.10 EXPLAIN ANALYZE

Syntax

EXPLAIN ANALYZE [VERBOSE] statement

Description

This statement is used to execute an SQL statement and display the distributed execution plan and the cost of each operation in the process.

VERBOSE is optional. If this parameter is specified, more detailed information and bottom-layer statistics are displayed. The statistics may not be accurate, especially for statements that are executed quickly.

Remarks

EXPLAIN ANALYZE does not support DDL statements.

Example

In the following example, you can view the CPU time consumed by each stage and the cost of each plan node.

NOTICE

The cost is based on the actual time (wall time) instead of the CPU-related time.

For each plan node, you can see additional statistics, such as the average input value of each node instance and the average number of hash collisions. The statistics are useful for analyzing data exceptions (such as data skewness and abnormal hash collisions) in an SQL statement.

```
EXPLAIN  ANALYZE  SELECT count(*),sum(totalprice) FROM new_orders GROUP BY orderstatus;
                                          Query Plan
-----------------------------------------------------------------------------------------------------------------------------------------
-
 Fragment 1 [HASH]
    CPU: 29.19ms, Scheduled: 134.78ms, Input: 2 rows (77B); per task: avg.: 1.00 std.dev.: 1.00, Output: 2
rows (36B)
    Output layout: [count, sum]
    Output partitioning: SINGLE []
    Stage Execution Strategy: UNGROUPED_EXECUTION
    Project[]
    │   Layout: [count:bigint, sum:double]
    │   Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
    │   CPU: 4.00ms (2.34%), Scheduled: 10.00ms (33.33%), Output: 2 rows
(36B)
    │   Input avg.: 0.06 rows, Input std.dev.: 387.30%
    └─ Aggregate(FINAL)[orderstatus][$hashvalue]
       │   Layout: [orderstatus:varchar, $hashvalue:bigint, count:bigint,
sum:double]
       │   Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
       │   CPU: 6.00ms (3.51%), Scheduled: 17.00ms (56.67%), Output: 2 rows
(77B)
       │   Input avg.: 0.06 rows, Input std.dev.: 387.30%
       │   count := count("count_9")
       │   sum := sum("sum_10")
       └─ LocalExchange[HASH][$hashvalue] ("orderstatus")
          │   Layout: [orderstatus:varchar, sum_10:double, count_9:bigint,
$hashvalue:bigint]
          │   Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
          │   CPU: 2.00ms (1.17%), Scheduled: 3.00ms (10.00%), Output: 2 rows
(77B)
          │   Input avg.: 0.06 rows, Input std.dev.: 556.78%
          └─ RemoteSource[2]
               Layout: [orderstatus:varchar, sum_10:double, count_9:bigint,
```



```
$hashvalue_11:bigint]
                CPU: 1.00ms (0.58%), Scheduled: 3.00ms (10.00%), Output: 2 rows
(77B)
                Input avg.: 0.06 rows, Input std.dev.: 556.78%

 Fragment 2 [SOURCE]
    CPU: 17.35ms, Scheduled: 80.04ms, Input: 4 rows (81B); per task: avg.: 4.00 std.dev.: 0.00, Output: 2 rows
(77B)
    Output layout: [orderstatus, sum_10, count_9, $hashvalue_12]
    Output partitioning: HASH [orderstatus][$hashvalue_12]
    Stage Execution Strategy: UNGROUPED_EXECUTION
    Aggregate(PARTIAL)[orderstatus][$hashvalue_12]
    │   Layout: [orderstatus:varchar, $hashvalue_12:bigint, sum_10:double,
count_9:bigint]
    │   CPU: 1.00ms (0.58%), Scheduled: 6.00ms (20.00%), Output: 2 rows
(77B)
    │   Input avg.: 4.00 rows, Input std.dev.: 0.00%
    │   sum_10 := sum("totalprice")
    │   count_9 := count(*)
    └─ ScanProject[table = hive:default:new_orders, grouped = false]
          Layout: [orderstatus:varchar, totalprice:double, $hashvalue_12:bigint]
          Estimates: {rows: 4 (292B), cpu: 256, memory: 0B, network: 0B}/{rows: 4 (292B), cpu: 548, memory:
0B, network: 0B}
          CPU: 16.00ms (9.36%), Scheduled: 132.00ms (440.00%), Output: 4 rows
(117B)
          Input avg.: 4.00 rows, Input std.dev.: 0.00%
          $hashvalue_12 := "combine_hash"(bigint '0', COALESCE("$operator$hash_code"("orderstatus"),
0))
          orderstatus := orderstatus:string:1:REGULAR
          totalprice := totalprice:double:2:REGULAR
          Input: 4 rows (81B), Filtered: 0.00%
(1 row)
```

11.13.7.11 REFRESH CATALOG

This statement is used to manually refresh the HetuEngine MetaStore cache to synchronize the metadata of tables, partitions, and databases of the Hive data source.

Syntax

REFRESH CATALOG catalog_name

Example

Log in to FusionInsight Manager, choose **Services** > **HetuEngine** > **Dashboard**. On the page that is displayed, click the link next to **HSConsole WebUI** to go to the compute instance page. On the displayed page, choose **Data Source** > *Hive data source name* > **Edit**. On the **Custom Configuration** page, click **Add** to add the following custom configuration items:

Parameter	Value	Description
hive.metastore-cache-ttl	5m	Cache validity period, in minutes
hive.metastore-refresh-interval	5m	Interval for refreshing the metadata cache, in minutes

Use Hive to create a table **tb3** and the query result on Hetu-cli is:

```
 show tables;
 Table
-------
 tb1
 tb2
(2 rows)
```

Refresh the metadata cache and query again.

```
 refresh catalog hive;
show tables;
 Table
-------
 tb1
 tb2
 tb3
(3 rows)
```

11.13.7.12 REFRESH SCHEMA

- Syntax

  REFRESH SCHEMA schema_name

- Description

  This statement is used to refresh the schema metadata cache.

- Example

  ```
  refresh schema default;
  REFRESH
  ```

11.13.7.13 REFRESH TABLE

- Syntax

  REFRESH TABLE table_name

- Description

  This statement is used to refresh the table metadata cache.

- Example

  ```
  refresh table fruit;
  REFRESH
  ```

11.13.7.14 ANALYZE

Syntax

ANALYZE table_name [WITH (property_name = expression [, ...])]

Description

This statement is used to collects statistics on tables and columns in a specified table.

The WITH clause is optional and can be used to specify connector properties. Run the **SELECT * FROM system.metadata.analyze_properties** command to list all available properties. Currently, only the Hive connector supports this property.

Example

- Collect the statistics of table **fruit**:
  ```
  ANALYZE fruit;
  ```
- To collect statistics on table storage in **catalog hive** and **schema default**:
  ```
  ANALYZE hive.default.orders;
  ```
- Collect information about the **2020-07-17** and **2020-07-18** partitions from the Hive partition table:
  ```
  ANALYZE hive.web.page_views WITH (partitions = ARRAY[ARRAY['2020-07-17','US'],
  ARRAY['2020-07-18','US']]);
  ```

11.13.7.15 CALL

Syntax

```
CALL procedure_name ( [ name => ] expression [, ...] )
```

Description

This statement is used to invoke a specified stored procedure.

The stored procedure is provided by each connector to implement data operations or management tasks. For example, a system connector allows a stored procedure to cancel a running query. Some data sources, such as PostgreSQL, have their own stored procedures, which are different from those defined by the connector and cannot be invoked by CALL.

Check and update the partition arrays in MetaStore. There are three modes.

- ADD: synchronizes the partition system that exists in the file system but does not exist in MetaStore to MetaStore.
- DROP: drops the partitions that exist in the metadata table but do not exist in the file system.
- FULL: performs both ADD and DROP operations.

Example

```
CALL system.create_empty_partition(
    schema_name => 'web',
    table_name => 'page_views',
    partition_columns => ARRAY['ds', 'country'],
    partition_values => ARRAY['2020-07-19', 'UK']);
```

Stored procedure supported by Hive Connector.

```
system.sync_partition_metadata(schema_name, table_name, mode)
```

11.13.7.16 PREPARE

Syntax

```
PREPARE statement_name FROM statement
```

Description

This statement is used to preprocess a statement for later execution. A preprocessing statement is used to save a query in a session with a specified

name. A statement can contain parameters to replace the text to be replaced during execution. The parameters are represented by question marks (?).

Example

- To preprocess a query:
  ```
  PREPARE my_select1 FROM SELECT * FROM fruit;
  ```
- The following shows how to preprocess a query that contains parameters. In **EXECUTE**, the values compared with **regionkey** and **nationkey** are replaced.
  ```
  PREPARE my_select2 FROM  SELECT name FROM fruit WHERE name= ? AND  price< ?;
  ```
- To preprocess INSERT query:
  ```
  PREPARE my_insert FROM INSERT INTO fruit VALUES ('watermelon',18);
  ```

11.13.7.17 DEALLOCATE PREPARE

Syntax

DEALLOCATE PREPARE statement_name

Description

This statement is used to remove the statement named **statement_name** from the preprocessing statement list in the session.

Example

To delete the preprocessing statement **name my_query**:

```
DEALLOCATE PREPARE my_select1;
```

11.13.7.18 EXECUTE

Syntax

EXECUTE statement_name [USING parameter1 [, parameter2, ...]]

Description

This statement is used to run the prepared SQL statement and use USING to specify input parameters.

Example

- To run a precompiled statement without input parameters.
  ```
  PREPARE my_select1 FROM SELECT name FROM fruit;
  EXECUTE my_select1;
  ```
- To run the SQL statement with two input parameters:
  ```
  PREPARE my_select2 FROM SELECT name FROM  fruit WHERE name= ? and price< ?;
  EXECUTE my_select2 USING  'peach',10;
  This is equivalent to:
  SELECT name FROM fruit WHERE name = 'peach' AND price<10;
  ```


11.13.7.19 VERIFY

Syntax

VERIFY MATERIALIZED VIEW MVNAME (mvname1,mvname2...) ORIGINALSQL query

Description

This statement is used to verify that whether a given SQL query statement can be rewritten by the specified materialized view.

Example

Verify that a specified SQL statement can be rewritten by materialized views **mv.tpcds.test** and **mv.tpcds.t1**.

```
verify materialized view mvname(mv.tpcds.test,mv.tpcds.t1) originalsql select c1 from t1 where id < 7;
MV_NAME    | VERIFY RESULT |        REMARKS
---------------|---------------|-------------------------
 mv.tpcds.test | true          | MV verified
 mv.tpcds.t1   | false         | This MV is not present
(2 rows)
```

11.13.8 HetuEngine Reserved Keywords

Table 11-79 lists the keywords reserved by the system and whether they are reserved in other SQL standards. If you need to use these keywords as identifiers, add double quotation marks.

Table 11-79 Keywords

Keyword	SQL:2016	SQL-92
ALTER	reserved	reserved
AND	reserved	reserved
AS	reserved	reserved
BETWEEN	reserved	reserved
BY	reserved	reserved
CASE	reserved	reserved
CAST	reserved	reserved
CONSTRAINT	reserved	reserved
CREATE	reserved	reserved
CROSS	reserved	reserved
CUBE	reserved	reserved
CURRENT_DATE	reserved	reserved

Keyword	SQL:2016	SQL-92
CURRENT_PATH	reserved	reserved
CURRENT_ROLE	reserved	reserved
CURRENT_TIME	reserved	reserved
CURRENT_TIMESTAMP	reserved	reserved
CURRENT_USER	reserved	reserved
DEALLOCATE	reserved	reserved
DELETE	reserved	reserved
DESCRIBE	reserved	reserved
DISTINCT	reserved	reserved
DROP	reserved	reserved
ELSE	reserved	reserved
END	reserved	reserved
ESCAPE	reserved	reserved
EXCEPT	reserved	reserved
EXECUTE	reserved	reserved
EXISTS	reserved	reserved
EXTRACT	reserved	reserved
FALSE	reserved	reserved
FOR	reserved	reserved
FROM	reserved	reserved
FULL	reserved	reserved
GROUP	reserved	reserved
GROUPING	reserved	reserved
HAVING	reserved	reserved
IN	reserved	reserved
INNER	reserved	reserved
INSERT	reserved	reserved
INTERSECT	reserved	reserved
INTO	reserved	reserved
IS	reserved	reserved

Keyword	SQL:2016	SQL-92
JOIN	reserved	reserved
LEFT	reserved	reserved
LIKE	reserved	reserved
LOCALTIME	reserved	reserved
LOCALTIMESTAMP	reserved	reserved
NATURAL	reserved	reserved
NORMALIZE	reserved	reserved
NOT	reserved	reserved
NULL	reserved	reserved
ON	reserved	reserved
OR	reserved	reserved
ORDER	reserved	reserved
OUTER	reserved	reserved
PREPARE	reserved	reserved
RECURSIVE	reserved	reserved
RIGHT	reserved	reserved
ROLLUP	reserved	reserved
SELECT	reserved	reserved
TABLE	reserved	reserved
THEN	reserved	reserved
TRUE	reserved	reserved
UESCAPE	reserved	reserved
UNION	reserved	reserved
UNNEST	reserved	reserved
USING	reserved	reserved
VALUES	reserved	reserved
WHEN	reserved	reserved
WHERE	reserved	reserved
WITH	reserved	reserved

11.13.9 HetuEngine Implicit Data Type Conversion

11.13.9.1 Enabling HetuEngine Implicit Data Type Conversion

HetuEngine automatically converts types when the queried data types do not match the data types of the table when users access the HetuEngine resources through the client. This avoids inconvenience caused by strong data type verification. Currently, the data type can be implicitly converted for INSERT statement, WHERE conditions, operations (+, -, *, and /), and function calls (connection operations ||).

The implicit type conversion function can be enabled or disabled. By default, the function is disabled. Before using the function, you need to enable it.

Enabling Implicit Conversion at the Session Level

Step 1 Log in to the HetuEngine client.

Step 2 Run the following command to enable implicit data type conversion:

set session implicit_conversion=true;

----End

Enabling Implicit Conversion of UDF Calculation Results at the Session Level

Step 1 Log in to the HetuEngine client.

Step 2 Run the following command to enable implicit conversion of UDF calculation results:

set session udf_implicit_conversion=true;

----End

Enabling Implicit Conversion or Implicit Conversion of UDF Calculation Results at the System Level

Step 1 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HetuEngine** > **Dashboard**, and click the HSConsole link on the **HSConsole Web UI** to go to the compute instance page.

Step 2 On the **Compute Instance** page that is displayed, select the target compute instance in the instance list, click **Stop** above the list to stop this instance, and click **Configure** in the **Operation** column to access its configuration page.

Step 3 Add the following custom parameters based on the application scenario and save the configuration:

Parameter	Value	Configuration File	Description
implicit-conversion	true	coordinator.config.properties	Implicit conversion

Parameter	Value	Configuration File	Description
udf-implicit-conversion	true	coordinator.config.properties	Implicit conversion of UDF calculation results

Step 4 Select **Start Now** and click **OK**.

----End

11.13.9.2 Disabling HetuEngine Implicit Data Type Conversion

Disabling Implicit Conversion at the Session Level

Step 1 Log in to the HetuEngine client.

Step 2 Run the following command to disable the implicit conversion function:

set session implicit_conversion=false;

----End

Disabling Implicit Conversion of UDF Calculation Results at the Session Level

Step 1 Log in to the HetuEngine client.

Step 2 Run the following command to disable the implicit conversion function:

set session udf_implicit_conversion=false;

----End

Disabling Implicit Conversion or Implicit Conversion of UDF Calculation Results at the System Level

Step 1 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HetuEngine** > **Dashboard**, and click the HSConsole link on the **HSConsole Web UI** to go to the compute instance page.

Step 2 On the **Compute Instance** page that is displayed, select the target compute instance in the instance list, click **Stop** above the list to stop this instance, and click **Configure** in the **Operation** column to access its configuration page.

Step 3 Delete the following custom parameters based on the application scenario and save the configuration:

Parameter	Value	Configuration File	Description
implicit-conversion	true	coordinator.config.properties	Implicit conversion

Parameter	Value	Configuration File	Description
udf-implicit-conversion	true	coordinator.config.properties	Implicit conversion of UDF calculation results

Step 4 Select **Start Now** and click **OK**.

----End

11.13.9.3 HetuEngine Implicit Conversion Table

After the implicit conversion function is enabled, implicit conversion is performed when the data type does not match. However, not all data types support implicit conversion. The following table lists the data type conversion tables supported by the implicit conversion function.

Table 11-80 Implicit conversion table

-	BOOLEAN	TINYINT	SMALLINT	INTEGER	BIGINT	REAL	DOUBLE	DECIMAL	VARCHAR
BOOLEAN	\	Y(1)	Y	Y	Y	Y	Y	Y	Y(2)
TINYINT	Y(3)	\	Y	Y	Y	Y	Y	Y	Y
SMALLINT	Y	Y(4)	\	Y	Y	Y	Y	Y	Y
INTEGER	Y	Y	Y	\	Y	Y	Y	Y	Y
BIGINT	Y	Y	Y	Y	\	Y	Y	Y	Y
REAL	Y	Y	Y	Y	Y	\	Y	Y(5)	Y
DOUBLE	Y	Y	Y	Y	Y	Y	\	Y	Y
DECIMAL	Y	Y	Y	Y	Y	Y	Y	\(6)	Y
VARCHAR	Y(7)	Y	Y	Y	Y	Y	Y	Y(8)	\
CHAR	N	N	N	N	N	N	N	N	Y
VARBINARY	N	N	N	N	N	N	N	N	N
JSON	N	N	N	N	N	N	N	N	Y

-	BOOLEAN	TINYINT	SMALLINT	INTEGER	BIGINT	REAL	DOUBLE	DECIMAL	VARCHAR
DATE	N	N	N	N	N	N	N	N	Y
TIME	N	N	N	N	N	N	N	N	Y
TIME WITH TIME ZONE	N	N	N	N	N	N	N	N	Y
TIMESTAMP	N	N	N	N	N	N	N	N	Y
TIMESTAMP WITH TIME ZONE	N	N	N	N	N	N	N	N	Y

Table 11-81 Implicit conversion table (continued)

-	CHAR	VARBINARY	JSON	DATE	TIME	TIME WITH TIME ZONE	TIMESTAMP	TIMESTAMP WITH TIME ZONE
BOOLEAN	N	N	Y	N	N	N	N	N
TINYINT	N	N	Y	N	N	N	N	N
SMALLINT	N	N	Y	N	N	N	N	N
INTEGER	N	N	Y	N	N	N	N	N
BIGINT	N	N	Y	N	N	N	N	N
REAL	N	N	Y	N	N	N	N	N
DOUBLE	N	N	Y	N	N	N	N	N
DECIMAL	N	N	Y	N	N	N	N	N

-	CHAR	VARBINARY	JSON	DATE	TIME	TIME WITH TIME ZONE	TIMESTAMP	TIMESTAMP WITH TIME ZONE
VARCHAR	Y(9)	Y	Y	Y(10)	Y(11)	Y(12)	Y(13)	Y
CHAR	\	N	N	N	N	N	N	N
VARBINARY	N	\	N	N	N	N	N	N
JSON	N	N	\	N	N	N	N	N
DATE	N	N	Y	\	N	N	Y(14)	Y
TIME	N	N	N	N	\	Y(15)	Y(16)	Y
TIME WITH TIME ZONE	N	N	N	N	Y	\	Y	Y
TIMESTAMP	N	N	N	Y	Y	Y	\	Y
TIMESTAMP WITH TIME ZONE	N	N	N	Y	Y	Y	Y	\

NOTE

1. The result of BOOLEAN->NUMBER can only be **0** or **1**.
2. The result of BOOLEAN->VARCHAR can only be **TRUE** or **FALSE**.
3. For NUMBER -> BOOLEAN, **0** is **false**, and other values are **true**.
4. The value of BIG PRECISION -> SMALL cannot be greater than the value range of the target type. Otherwise, an error is reported.
5. The integer part of REAL/FLOAT ->DECIMAL must be greater than or equal to the integer part of REAL/FLOAT. Otherwise, an error is reported during conversion. If the decimal part is insufficient, the data is truncated.
6. The integer part of the DECIMAL->DECIMAL target type must be greater than or equal to that of the source type. Otherwise, the conversion fails and the decimal part will be truncated if it is insufficient.
7. For VARCHAR->BOOLEAN, only **0**, **1**, **TRUE**, and **FALSE** can be converted.
8. For VARCHAR->DECIMAL, if the number of decimal places is greater than the number of decimal places of the target decimal, truncation occurs. If the number of integer places is greater than the number of decimal places of the target decimal, an error is reported.
9. For VARCHAR->CHAR, if the length of VARCHAR exceeds the target length, truncation occurs.
10. For VARCHAR->DATE, dates can only be separated by hyphens (-), for example, 2000-01-01.
11. For VARCHAR->TIME, only the strict date format HH:MM:SS.XXX is supported.
12. For VARCHAR->TIME ZONE, only the strict time format is supported, for example, 01:02:03.456 America/Los_Angeles.
13. For VARCHAR->TIMESTAMP, only the strict format YYYY-MM-DD HH:MM:SS.XXX is supported.
14. DATE->TIMESTAMP automatically supplements the time by adding 0s to the end, for example, 2010-01-01 to 2010-01-01 00:00:00.000.
15. TIME->TIME WITH TIME ZONE automatically supplements the time zone.
16. TIME->TIMESTAMP automatically supplements the date. The default value **1970-01-01** is used.

11.13.10 Data Preparation for the Sample Table

Create a table containing TINYINT data

```
-- Create a table containing TINYINT data:
 CREATE TABLE int_type_t1   (IT_COL1 TINYINT) ;
--Insert data of the TINYINT type.
 insert into int_type_t1  values (TINYINT'10');
-- Create a table containing DECIMAL data:
CREATE TABLE decimal_t1       (dec_col1 DECIMAL(10,3)) ;
– Insert data of the DECIMAL type:
insert into decimal_t1  values (DECIMAL '5.325' );
create table array_tb(col1 array<int>,col2 array<array<int>>);
create table row_tb(col1 row(a int,b varchar));
```

Create a Map table

```
-- Create a Map table.
create table map_tb(col1 MAP<STRING,INT>);
-- Insert a piece of data of the Map type.
insert into map_tb values(MAP(ARRAY['foo','bar'],ARRAY[1,2]));
--Query data.
select * from map_tb; --   {bar=2, foo=1}
```

Create a ROW table


```
-- Create a ROW table.
create table row_tb (id int,col1 row(a int,b varchar));
-- Insert data of the ROW type.
insert into row_tb values (1,ROW(1,'SSS'));
--Query data.
select * from row_tb; --
 id |     col1
----|--------------
  1 | {a=1, b=SSS}
select col1.b from row_tb; -- SSS
select col1[1] from row_tb; -- 1
```

Creating a STRUCT table

```
-- Creating a STRUCT table.
create table struct_tab (id int,col1 struct<col2: integer, col3: string>);
-- Insert data of the STRUCT type.
 insert into struct_tab VALUES(1, struct<2, 'test'>);
--Query data.
select * from struct_tab; --
 id |         col1
----|----------------------
  1 | {col2=2, col3=test}
```

Creating a schema named "web"

```
-- Creating a schema named "web":
CREATE SCHEMA web;
--Create a schema named sales in the Hive data source:
CREATE SCHEMA hive.sales;
-- Creating a schema named traffic, if it does not exist:
CREATE SCHEMA IF NOT EXISTS traffic;
```

Create a new table **orders** and use the WITH clause to specify the storage format, storage location, and whether the table is an external table

```
-- Create a new table orders and use the WITH clause to specify the storage format, storage location, and
whether the table is an external table.
CREATE TABLE orders (
orderkey bigint,
orderstatus varchar,
totalprice double,
orderdate date
)
WITH (format = 'ORC', location='/user',external=true);
-- If the orders table does not exist, create the orders table and add table comments and column
comments:
CREATE TABLE IF NOT EXISTS new_orders (
orderkey bigint,
orderstatus varchar,
totalprice double COMMENT 'Price in cents.',
orderdate date
)
COMMENT 'A table to keep track of orders.';
-- Create the bigger_orders table using the column definition of the orders table:
CREATE TABLE bigger_orders (
another_orderkey bigint,
LIKE orders,
another_orderdate date
);

CREATE SCHEMA hive.web WITH (location = 'hdfs://hacluster/user');
--Create a partitioned table.
CREATE TABLE hive.web.page_views (
  view_time timestamp,
  user_id bigint,
  page_url varchar,
  ds date,
  country varchar
)
```



```
WITH (
  format = 'ORC',
  partitioned_by = ARRAY['ds', 'country'],
  bucketed_by = ARRAY['user_id'],
  bucket_count = 50
);
--Insert an empty partition.
CALL system.create_empty_partition(
    schema_name => 'web',
    table_name => 'page_views',
    partition_columns => ARRAY['ds', 'country'],
    partition_values => ARRAY['2020-07-17', 'US']);

CALL system.create_empty_partition(
    schema_name => 'web',
    table_name => 'page_views',
    partition_columns => ARRAY['ds', 'country'],
    partition_values => ARRAY['2020-07-18', 'US']);
--Insert data.
insert into hive.web.page_views values(timestamp '2020-07-17 23:00:15',bigint '15141','www.local.com',date
'2020-07-17','US' );
insert into hive.web.page_views values(timestamp '2020-07-17 23:00:16',bigint '15142','www.abc.com',date
'2020-07-17','US' );
insert into hive.web.page_views values(timestamp '2020-07-18 23:00:18',bigint '18148','www.local.com',date
'2020-07-18','US' );

-- Delete all data in the partition specified by the WHERE clause from the partitioned table.
delete from hive.web.page_views where ds=date '2020-07-17' and country='US';

--Run the following statement to create the orders_column_aliased table based on the query result of a
specified column:
CREATE TABLE orders_column_aliased (order_date, total_price)
AS
SELECT orderdate, totalprice FROM orders;
--Create the orders_by_data table based on the summary result of the orders table.
CREATE TABLE orders_by_date
COMMENT 'Summary of orders by date'
WITH (format = 'ORC')
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
--If the orders_by_date table does not exist, create the orders_by_date table:
CREATE TABLE IF NOT EXISTS orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
--Create the empty_orders table using the schema that is the same as that of the orders table but does
not contain data.
CREATE TABLE empty_orders AS
SELECT *
FROM orders
WITH NO DATA;
```

Create a view named **test_view** in the **orders** table

```
--Create a view named test_view in the orders table:
CREATE VIEW test_view (oderkey comment 'orderId',orderstatus comment 'status',half comment 'half') AS
SELECT orderkey, orderstatus, totalprice / 2 AS half FROM orders;
--Create the orders_by_date_view view based on the summary result of the orders table.
CREATE VIEW orders_by_date_view AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
--Create a new view to replace the existing view:
CREATE OR REPLACE VIEW test_view AS
SELECT orderkey, orderstatus, totalprice / 4 AS quarter
FROM orders;
```

Modify the definition of an existing table.


```
--Data preparation.
create table users (id int,name varchar);
--Change the table name from users to people:
ALTER TABLE users RENAME TO people;
--Add the zip column to the people table:
ALTER TABLE people ADD COLUMN zip varchar;
--Delete the zip column from the people table:
ALTER TABLE people DROP COLUMN zip;
--change the column name id in the people table to user_id:
ALTER TABLE people RENAME COLUMN id TO user_id;

create table testfordrop(name varchar);
```

Other

```
--CreatE a view.
create view orders_by_date as select * from orders;
--Set the comment information of a table. You can delete the comment by setting the comment information to NULL.
COMMENT ON TABLE people IS 'master table';

--create a table with the column id and name:
CREATE TABLE example AS
SELECT * FROM (
VALUES
(1, 'a'),
(2, 'b'),
(3, 'c')
) AS t (id, name);

--Create the fruit and fruit_copy tables:
create table fruit (name varchar,price double);
create table fruit_copy (name varchar,price double);
--Insert a row of data into the fruit table:
insert into fruit values('LIchee',32);
--Insert multiple lines of data into the fruit table:
insert into fruit values('banana',10),('peach',6),('lemon',12),('apple',7);
--Load the data lines in the fruit table to the fruit_copy table. After the execution, there are five records in the table.
insert into fruit_copy select * from fruit;
--Clear the fruit_copy table, and then load the data in the fruit table to the table. After the execution, there are two records in the fruit_copy table.
insert overwrite fruit_copy select *  from fruit limit 2;

--Create a shipping table:
create table shipping(origin_state varchar(25),origin_zip integer,destination_state
varchar(25) ,destination_zip integer,package_weight integer);

--Insert data.
insert into shipping values ('California',94131,'New Jersey',8648,13),
('California',94131,'New Jersey',8540,42),
('California',90210,'Connecticut',6927,1337),
('California',94131,'Colorado',80302,5),
('New York',10002,'New Jersey',8540,3),
('New Jersey',7081,'Connecticut',6708,225);

--Create a table and insert data into the table.
create table cookies_log (cookieid varchar,createtime  date,pv int);
insert into cookies_log values
    ('cookie1',date '2020-07-10',1),
    ('cookie1',date '2020-07-11',5),
    ('cookie1',date '2020-07-12',7),
    ('cookie1',date '2020-07-13',3),
    ('cookie1',date '2020-07-14',2),
    ('cookie1',date '2020-07-15',4),
    ('cookie1',date '2020-07-16',4),
    ('cookie2',date '2020-07-10',2),
    ('cookie2',date '2020-07-11',3),
    ('cookie2',date '2020-07-12',5),
     ('cookie2',date '2020-07-13',6),
```



```
    ('cookie2',date '2020-07-14',3),
    ('cookie2',date '2020-07-15',9),
  ('cookie2',date '2020-07-16',7);

--Create a table.
create  table new_shipping (origin_state varchar,origin_zip varchar,packages int ,total_cost int);

--Insert data.
insert into new_shipping
values
('California','94131',25,100),
('California','P332a',5,72),
('California','94025',0,155),
('New Jersey','08544',225,490);

--Create a data table and insert data.
create table salary (dept varchar, userid varchar, sal double);
 insert into salary values ('d1','user1',1000),('d1','user2',2000),('d1','user3',3000),('d2','user4',4000),
('d2','user5',5000);

-- Prepare data.
create table cookie_views( cookieid varchar,createtime timestamp,url varchar);
insert into cookie_views values
('cookie1',timestamp '2020-07-10 10:00:02','url20'),
('cookie1',timestamp '2020-07-10 10:00:00','url10'),
('cookie1',timestamp '2020-07-10 10:03:04','urll3'),
('cookie1',timestamp '2020-07-10 10:50:05','url60'),
('cookie1',timestamp '2020-07-10 11:00:00','url70'),
('cookie1',timestamp '2020-07-10 10:10:00','url40'),
('cookie1',timestamp '2020-07-10 10:50:01','url50'),
('cookie2',timestamp '2020-07-10 10:00:02','url23'),
('cookie2',timestamp '2020-07-10 10:00:00','url11'),
('cookie2',timestamp '2020-07-10 10:03:04','url33'),
('cookie2',timestamp '2020-07-10 10:50:05','url66'),
('cookie2',timestamp '2020-07-10 11:00:00','url77'),
('cookie2',timestamp '2020-07-10 10:10:00','url47'),
('cookie2',timestamp '2020-07-10 10:50:01','url55');

CREATE TABLE visit_summaries (  visit_date date,  hll varbinary);

insert into visit_summaries select createtime,cast(approx_set(cookieid) as varbinary) from cookies_log
group by createtime;

CREATE TABLE nation (name varchar,  regionkey integer);
insert into nation values ('ETHIOPIA',0),
('MOROCCO',0),
('ETHIOPIA',0),
('KENYA',0),
('ALGERIA',0),
('MOZAMBIQUE',0);

CREATE TABLE region (  name varchar,  regionkey integer);
insert into region values ('ETHIOPIA',0),
('MOROCCO',0),
('ETHIOPIA',0),
('KENYA',0),
('ALGERIA',0),
('MOZAMBIQUE',0);
```


11.13.11 HetuEngine Syntax Compatibility with Common Data Sources

Syntax	Hive	MPPDB	Elasticsearch	HBase	HetuEngine (Cross-domain)	ClickHouse	Hudi	MySQL
show schemas of databases	Y	Y	Y	Y	Y	Y	Y	Y
create schema of databases	Y	Y	N	Y	N	N	Y	N
use schema of databases	Y	Y	Y	Y	Y	Y	Y	Y
alter schema of databases	Y	N	N	N	N	N	N	N
drop schema of databases	Y	Y	Y	Y	N	N	Y	N
show tables/show create table/show functions/show session of tables	Y	Y	Y	Y	Y	Y	Y	Y
create of tables	Y	Y	N	Y	N	N	N	N
create table TABLENAME as of tables	Y	Y	Y	Y	N	N	N	N
insert into TABLENAME values of tables	Y	Y	Y	Y	Y	N	N	N
insert into TABLENAME select of tables	Y	Y	Y	Y	Y	N	N	N
insert overwrite TABLENAME values of tables	Y	N	N	N	N	N	N	N
insert overwrite TABLENAME select of tables	Y	N	N	N	N	N	N	N
alter of tables	Y	Y	N	N	N	N	N	N
select of tables	Y	Y	Y	Y	Y	Y	Y	Y
update of tables	Y	Y	Y	N	N	N	N	N

Syntax	Hive	MPPDB	Elasticsearch	HBase	HetuEngine (Cross-domain)	ClickHouse	Hudi	MySQL
delete of tables	Y	Y	Y	Y	N	N	N	N
drop of tables	Y	N	Y	Y	Y	N	N	N
desc/describe TABLENAME of tables	Y	Y	Y	Y	Y	Y	Y	Y
analyze of tables	Y	Y	Y	N	N	N	Y	N
comment of tables	Y	N	N	N	N	N	N	N
explain of tables	Y	Y	Y	Y	Y	N	Y	N
show stats of tables	Y	Y	Y	N	N	N	Y	N
show columns of tables	Y	Y	Y	Y	Y	Y	Y	Y
select column of tables	Y	Y	Y	Y	Y	Y	Y	Y
create view of views	Y	Y	N	N	N	N	N	N
create or replace view of views	Y	N	N	N	N	N	N	N
alter of views	Y	N	N	N	N	N	N	N
drop of views	Y	N	N	N	N	N	N	N
select of views	Y	Y	N	N	Y	Y	Y	Y
desc/describe VIEWNAME of views	Y	Y	N	N	Y	Y	Y	Y
show views/show create view of views	Y	Y	N	N	N	Y	Y	Y
show columns of views	Y	Y	Y	Y	Y	Y	Y	Y
select column of views	Y	Y	Y	Y	Y	Y	Y	Y

11.14 Common Issues About HetuEngine

11.14.1 What Should I Do After the HetuEngine Domain Name Is Changed?

Question

After the domain name is changed, the installed client configuration and data source configuration become invalid, and the created cluster is unavailable. When data sources in different domains are interconnected, HetuEngine automatically combines the **krb5.conf** file. After the domain name is changed, the domain name for Kerberos authentication changes. As a result, the information about the interconnected data source becomes invalid.

Answer

- You need to reinstall the cluster client.
- Delete the old data source information on HSConsole by referring to **Managing a HetuEngine Data Source**.
- Configure the data source information on HSConsole again by referring to **Adding a HetuEngine Data Source**.

11.14.2 What Can I Do If Starting the HetuEngine Cluster on the Client Times Out?

Question

If the cluster startup on the client takes a long time, the waiting times out and the waiting page exits.

Answer

If the cluster startup times out, the waiting page automatically exits.

You can log in to the cluster again until the cluster is successfully started.

Additionally, you can also view the cluster running status on the HSConsole page. When the cluster is in the running state, log in to the cluster again. If the cluster fails to be started, you can locate the fault based on the startup logs. For details, see **HetuEngine Log Overview**.

11.14.3 How Do I Handle Data Loss in a HetuEngine Data Source?

Question

Why is the data source lost when I log in to the client to check the data source connected to the HSConsole page?

Answer

The possible cause of data source loss is that the DBService active/standby switchover occurs or the database connection usage exceeds the threshold.

1. You can log in to FusionInsight Manager.
2. View the alarm information and clear the DBService alarm based on the alarm guide.

11.14.4 What Do I Do If the View Owner Does Not Have the Permission on Functions?

Question

When you access a view of an independently deployed Hive data source and a function is used in the Hive view, the error message "view owner does not have sufficient privileges" may be displayed.

Figure 11-12 Error message

Answer

The Ranger permission of HetuEngine controls function permissions. If the view owner of the independently deployed Hive data source cluster does not have a user with the same name in the cluster, the function permission configured in Ranger cannot be granted to the user. Perform the following operations to grant permission to the user and refer to **Adding a Ranger Access Permission Policy for HetuEngine**:

- Solution 1

  Create a user with the same name as the view owner in the cluster and modify the **all-function** policy in the Ranger access permission policy of HetuEngine.

  a. In the **Allow Condition** area, enter the authorized view owner in the **Select User** text box.

  b. In **Permissions**, select **Grant** and **execute**.

- Solution 2

  Modify the **all-function** policy in the Ranger access permission policy of HetuEngine to grant all users the following permissions:

  a. In the **Allow Condition** area, select **{USER}** from the **Select User** box.

  b. In **Permissions**, select **Grant** and **execute**.

  If you need to manage the function permission of a specific user, you can configure **Deny Conditions**.

  a. Select a user from **Select User**.

  b. In **Permissions**, select **Grant** and **execute** to disable the function permissions of the user.

11.14.5 What Do I Do If Error "Encountered too many errors" Is Reported During HetuEngine SQL Execution?

Question

The following error is reported during the execution of HetuEngine service SQL statements:

```
Encountered too many errors talking to a worker node. The node may have crashed or be under too much load. This is probably a transient issue, so please retry your query in a few minutes
```

Answer

Possible causes:

- Some Worker nodes are faulty, for example, the network is abnormal or the process memory usage is high.
- The worker node is overloaded, memory is used up (OOM), and the worker node cannot provide services.
- The worker node automatically restarts due to GC.

Solution:

- For faulty worker node:

  If errors are reported on the same worker node at different time, check the node. For example, check the network connection and memory usage.
- High worker node load and GC

  a. Log in to FusionInsight Manager as a user who can access the HetuEngine web UI and choose **Cluster** > **Services** > **HetuEngine**. The **HetuEngine** service page is displayed.

  b. Click **Dashboard**. In the **Basic Information** area, click the link next to **HSConsole WebUI**. The HSConsole page is displayed.

  c. On the **Compute Instances** page, expand the tenant to which the compute instance belongs and ensure that the instance you want to modify is **STOPPED**.

  d. Locate the row that contains the tenant to which the compute instance belongs, and click **Configure** in the **Operation** column. The **Configure Instance** tab is displayed.

     i. Increase the memory size of a worker in a compute instance.

        In the configuration of a single compute instance, increase the value of **Container Memory (MB)** of **Worker Container Resource Configuration** and the value of **-Xmx** in **JVM**.

     ii. Limit the memory usage of a query on a worker.

        In the **Custom Configuration** area, click **Add** to add the **query.max-memory-per-node** parameter twice for **coordinator.config.properties** and **worker.config.properties**, respectively. Set them to a value less than **70%** of the **-Xmx** in **JVM**.

  e. Click **OK** and restart the compute instance.

11.15 HetuEngine Troubleshooting

11.15.1 Python Not Exist When a HetuEngine Compute Instance Failed to Start

Question

HetuEngine compute instances fail to start, and the following error information is displayed in the **stderr.txt** file in a coordinator container:

```
/usr/bin/env: 'python': No such file or directory
```

Answer

The startup of HetuEngine compute instances depends on the Python file. Ensure that the Python file exists in the **/usr/bin/** directory on each node.

Step 1 Log in to FusionInsight Manager, choose **Hosts**, and view and record the service IP addresses of all hosts.

Step 2 Log in to the node recorded in **Step 1** as user **root** and run the following commands on all nodes to add the python3 soft link to the **/usr/bin/** directory:

cd /usr/bin

ln -s python3 python

Step 3 Restart the HetuEngine compute instance.

----End

11.15.2 HetuEngine Compute Instance Is Faulty After Being Started

Question

A HetuEngine compute instance enters the faulty state about 30 seconds after it is started.

Answer

When starting a compute instance, HetuEngine sends a command to Yarn to start the corresponding application. If HetuEngine does not receive a response from Yarn within 30 seconds, HetuEngine ends the request due to timeout.

If the response message for Yarn to start the application cannot be received within 30 seconds due to machine performance or network environment problems, you can prolong the timeout period.

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **HetuEngine** and click **Configurations** then **All Configurations**.

Step 3 Search for the **application.customized.properties** parameter, add the custom parameter **yarn.application.start.timeout**, set the timeout interval as required (enter only digits without the unit second), and save the configuration.

Step 4 Click the **Instance** tab, select all HSBroker instances, click **More**, and select **Restart Instance** to restart the HSBroker instances as prompted.

----End

12 Using Hive

12.1 Hive User Permission Management

12.1.1 About Hive User Permissions

Hive is a data warehouse framework built on Hadoop. It provides basic data analysis services using the Hive query language (HQL), a language like the structured query language (SQL).

MRS supports users, user groups, and roles. Permissions must be assigned to roles and then roles are bound to users or user groups. Users can obtain permissions only by binding a role or joining a group that is bound with a role. For details about Hive authorization, visit **https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Authorization**.

NOTE

- Hive permissions in security mode need to be managed whereas those in normal mode do not.
- If the current component uses Ranger for permission control, you need to configure permission management policies based on Ranger. For details, see **Adding a Ranger Access Permission Policy for Hive**.

Hive Permission Model

To use the Hive component, users must have permissions on Hive databases and tables (including external tables and views). In MRS, the complete Hive permission model is composed of Hive metadata permission and HDFS file permission. The Hive permission model also includes the permission to use databases or tables.

- Hive metadata permission

  Similar to traditional relational databases, the Hive database of MRS supports the **CREATE** and **SELECT** permission, and the Hive tables and columns support the **SELECT**, **INSERT**, and **DELETE** permissions. Hive also supports the permissions of **OWNERSHIP** and **Hive Admin Privilege**.
- Hive data file permission, also known as HDFS file permission

Hive database and table files are stored in the HDFS. The created databases or tables are saved in the **/user/hive/warehouse** directory of the HDFS by default. The system automatically creates subdirectories named after database names and database table names. To access a database or a table, the corresponding file permissions (read, write, and execute) on the HDFS are required.

To perform various operations on Hive databases or tables, you need to associate the metadata permission with the HDFS file permission. For example, to query Hive data tables, you need to associate the metadata permission **SELECT** and the HDFS file permissions **Read** and **Write**.

To use the role management function of Manager GUI to manage the permissions of Hive databases and tables, you only need to configure the metadata permission, and the system will automatically associate and configure the HDFS file permission. In this way, operations on the interface are simplified, and the efficiency is improved.

Hive Users

MRS provides users and roles to use Hive, such as creating tables, inserting data into tables, and querying tables. Hive defines the **USER** class, corresponding to user instances. Hive defines the **GROUP** class, corresponding to role instances.

You can use Manager to set permissions for Hive users. This method only supports permission setting in roles. A user or user group can obtain the permissions only after a role is bound to the user or user group. Hive users can be granted Hive administrator permissions and permissions to access databases, tables, and columns.

Support for Cascading Authorization (Available in MRS 3.3.0 or Later).

Hive tables in a cluster with Ranger authentication enabled support cascading authorization, which significantly improves the authentication usability. You only need to authorize for service tables once on the Ranger page, and the background automatically associates the permissions of the data storage source in a fine-grained manner without detecting the storage path of the tables and without requiring secondary authorization. This also eliminates the disadvantage of authorization based on decoupled storage and compute. For details, see **Hive Tables Supporting Cascading Authorization**.

Hive Usage Scenarios and Related Permissions

Creating a database with Hive requires users to join in the **hive** group, without granting a role. Users have all permissions on the databases or tables created by themselves in Hive or HDFS. They can create tables, select, delete, insert, or update data, and grant permissions to other users to allow them to access the tables and corresponding HDFS directories and files.

A user can access the tables or database only with permissions. The permission required by users varies according to Hive usage scenarios.

Table 12-1 Hive usage scenarios

Typical Scenario	Permission
Using Hive tables, columns, or databases	Permissions required in different scenarios are as follows: ● To create tables, the **CREATE** permission is required. ● To query data, the **SELECT** permission is required. ● To insert data, the **INSERT** permission is required. ● To delete data, the **DELETE** permission is required.
Associating and using other components	In addition to Hive permissions, permissions of other components are required in some scenarios, for example: ● Yarn permissions are required when some HQL statements, such as **insert**, **count**, **distinct**, **group by**, **order by**, **sort by**, and **join**, are run. You are advised to grant Yarn permissions to the role of each Hive user. ● HBase permission is required when Hive over HBase is used, for example, querying HBase table data in Hive.

In some special Hive usage scenarios, you need to configure other types of permission.

Table 12-2 Hive authorization precautions

Scenario	Permission
Creating Hive databases, tables, and external tables, or adding partitions to created Hive tables or external tables when data files specified by Hive users are saved to other HDFS directories except **/user/hive/warehouse**	The directory must already exist, the Hive user must be the owner of the directory, and the Hive user must have the read, write, and execute permissions on the directory. The user must have the **read** and **write** permissions of all the upper-layer directories of the directory. After an administrator grants the Hive permission to the role, the HDFS permission is automatically granted.

Scenario	Permission
Using **load** to load data from all the files or specified files in a specified directory to Hive tables as a Hive user	• The data source is a Linux local disk, the specified directory exists, and the system user **omm** has read and execute permission of the directory and all its upper-layer directories. The specified file exists, and user **omm** has read permission of the file and has the read and execute permission of all the upper-layer directories of the file. • The data source is HDFS, the specified directory exists, and the Hive user is the owner of the directory and has read, write, and execute permission on the directory and its subdirectories, and has read and write permission on all its upper-layer directories. The specified file exists, and the Hive user is the owner of the file and has read, write, and execute permission, and has read and execute permission on the file and all its upper-layer directories. **NOTE** When **load** is used to import data to a Linux local disk, files must be loaded to the HiveServer on which the command is run and the permission must be modified. You are advised to run the command on a client. The HiveSever to which the client is connected can be found. For example, if the Hive client displays **0: jdbc:hive2://10.172.0.43:21066/>**, the IP address of the connected HiveServer is 10.172.0.43.
Creating or deleting functions or modifying any database	The **Hive Admin Privilege** is required.
Performing operations on all databases and tables in Hive	The user must be added to the **supergroup** user group and granted **Hive Admin Privilege**.

Scenario	Permission
Enabling Ranger authentication when Kerberos authentication is disabled for the cluster (the cluster is in normal mode)	By default, Ranger authentication is disabled if Kerberos authentication is disabled for the cluster (the cluster is in normal mode). If Ranger authentication is disabled, there are the following restrictions: • Whitelist: If the whitelist function is enabled, you cannot set parameters on the client. You can disable the whitelist function by setting the **hive.security.whitelist.switch** parameter to **OFF** on the Hive configuration page. This poses security risks. Exercise caution when performing this operation. • In MRS 3.5.0 and later versions, the **transform** function cannot be executed. If required by services, add **hive.security.transform.disallow** to the HiveServer custom parameters, and set it to **false** to allow Hive to execute the **transform** function. This poses security risks. Exercise caution when performing this operation. • The **reflect**, **reflect2**, **java_method**, and **in_file** functions cannot be executed. If required, add **hive.server2.builtin.udf.blacklist** to the custom parameters of HiveServer and set it to **mpty_blacklist** to allow Hive to execute these functions. This poses security risks exist. Exercise caution when performing this operation.

12.1.2 Creating a Hive Role

Scenario

Create and configure a Hive role on Manager as an MRS cluster administrator. The Hive role can be granted the permissions of the Hive administrator and the permissions to operate Hive table data.

Creating a database with Hive requires users to join in the **hive** group, without granting a role. Users have all permissions on the databases or tables created by themselves in Hive or HDFS. They can create tables, select, delete, insert, or

update data, and grant permissions to other users to allow them to access the tables and corresponding HDFS directories and files. The created databases or tables are saved in the **/user/hive/warehouse** directory of the HDFS by default.

NOTE

- A Hive role can be created only in security mode.
- If the current component uses Ranger for permission control, you need to configure permission management policies based on Ranger. For details, see **Adding a Ranger Access Permission Policy for Hive**.

Prerequisites

- The MRS cluster administrator has understood service requirements.
- The Hive client has been installed.

Procedure

Step 1 Log in to FusionInsight Manager. For details, see**Accessing FusionInsight Manager**

Step 2 Choose **System** > **Permission** > **Role**.

Step 3 Click **Create Role**, and set **Role Name** and **Description**.

Step 4 Set **Configure Resource Permission**. For details, see **Table 12-3**.

- Grant the read and execution permissions for the HDFS directory.
  - Click *Name of the desired cluster* and select **HDFS** for **Service Name**. On the displayed page, click **File System**, choose **hdfs://hacluster/** > **user**, locate the row where **hive** is located, and select **Read** and **Execute** in the **Permission** column.
  - Click *Name of the desired cluster* and select **HDFS** for **Service Name**. On the displayed page, click **File System**, choose **hdfs://hacluster/** > **user** > **hive**, locate the row where **warehouse** is located, and select **Read** and **Execute** in the **Permission** column.
  - Click *Name of the desired cluster* and select **HDFS** for **Service Name**. On the displayed page, click **File System**, choose **hdfs://hacluster/** > **tmp**, locate the row where **hive-scratch** is located, and select **Read** and **Execute** in the **Permission** column.
- **Hive Admin Privilege**: Hive administrator permission.
- **Hive Read Write Privileges**: Hive data table management permission, which is the operation permission to set and manage the data of created tables.

NOTE

- Hive role management supports the administrator permissions and the permission of accessing databases, tables, and views.
- The permissions of the Hive administrator do not include the permission to manage HDFS.
- If there are too many tables in the database or too many files in tables, the permission granting may last a while. For example, if a table contains 10,000 files, the permission granting lasts about 2 minutes.

Table 12-3 Setting a role

Task	Role Authorization
Setting the Hive administrator permission	In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Hive** and select **Hive Admin Privilege**. **NOTE** After being bound to the Hive administrator role, perform the following operations during each maintenance operation: 1. Log in to the node where the Hive client is installed as the client installation user. 2. Run the following command to configure environment variables: For example, if the Hive client installation directory is **/opt/hiveclient**, run **source /opt/hiveclient/bigdata_env**. 3. Run the following command to authenticate the user: **kinit** *Hive service user* 4. Run the following command to log in to the client tool: **beeline** 5. Run the following command to update the administrator permissions of the Hive user: **set role admin;**
Setting the permission to query a table of another user in the default database	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Hive** > **Hive Read Write Privileges**. 2. Click the name of the specified database in the database list. Tables in the database are displayed. 3. In the **Rights** column of the specified table, choose **Select**.
Setting the permission to query a table of another user in the default database	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Hive** > **Hive Read Write Privileges**. 2. Click the name of the specified database in the database list. Tables in the database are displayed. 3. In the **Permission** column of the specified table, select **INSERT**.

Task	Role Authorization
Setting the permission to import data to a table of another user in the default database	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Hive** > **Hive Read Write Privileges**. 2. Click the name of the specified database in the database list. Tables in the database are displayed. 3. In the **Permission** column of the specified indexes, select **DELETE** and **INSERT**.
Setting the permission to submit HQL commands to Yarn for execution	The HQL commands used by some services are converted into MapReduce tasks and submitted to Yarn for execution. You need to set the Yarn permissions. For example, the HQL statements to be run use statements, such as **insert**, **count**, **distinct**, **group by**, **order by**, **sort by**, or **join**. 1. In the **Permission** table, choose *Name of the desired cluster* > **Yarn** > **Scheduling Queue** > **root**. 2. In the **Permission** column of the **default** queue, select **Submit**.

Step 5 Click **OK**, and return to the **Role** page.

----End

12.1.3 Granting Hive Permissions on Tables, Columns, or Databases

Scenario

You can configure related permissions if you need to access tables or databases created by other users. Hive supports column-based permission control. If a user needs to access some columns in tables created by other users, the user must be granted the permission for columns. The following describes how to grant table, column, and database permissions to users by using the role management function of MRS Manager.

NOTE

- You can configure permissions for Hive tables, columns, or databases only in security mode.
- If the current component uses Ranger for permission control, you need to configure permission management policies based on Ranger. For details, see **Adding a Ranger Access Permission Policy for Hive**.

Prerequisites

- You have obtained a user account with the administrator permissions, such as **admin**.
- You have created a role, for example, **hrole**, on Manager by referring to instructions in **Creating a Hive Role**. You do not need to set the Hive permission but need to set the permission to submit the HQL command to Yarn for execution.
- You have created two Hive human-machine users, such as **huser1** and **huser2**, on Manager and added them to the **hive** group. **huser2** has been bound to **hrole**. The **hdb** database has created by user **huser1** and the **htable** table has been created in the database.

Procedure

- Granting Table Permissions

  Users have complete permission on the tables created by themselves in Hive and the HDFS. To access the tables created by others, they need to be granted the permission. After the Hive metadata permission is granted, the HDFS permission is automatically granted. The procedure for granting a role the permission of querying, inserting, and deleting **htable** data is as follows:

  a. On FusionInsight Manager, choose **System** > **Permission** > **Role**.

  b. Locate the row that contains **hrole**, and click **Modify**.

  c. Choose *Name of the desired cluster* > **Hive** > **Hive Read Write Privileges**.

  d. Click the name of the specified database **hdb** in the database list. Table **htable** in the database is displayed.

  e. In the **Permission** column of the **htable** table, select **SELECT**, **INSERT**, and **DELETE**.

  f. Click **OK**.

  NOTE

  In role management, the procedure for granting a role the permission of querying, inserting, and deleting Hive external table data is the same. After the metadata permission is granted, the HDFS permission is automatically granted.

- Granting Column Permissions

  Users have all permissions for the tables created by themselves in Hive and HDFS. Users do not have the permission to access the tables created by others. If a user needs to access some columns in tables created by other users, the user must be granted the permission for columns. After the Hive

metadata permission is granted, the HDFS permission is automatically granted. The procedure for granting a role the permission of querying and inserting data in **hcol** of **htable** is as follows:

a. On FusionInsight Manager, choose **System** > **Permission** > **Role**.

b. Locate the row that contains **hrole**, and click **Modify**.

c. Choose *Name of the desired cluster* > **Hive** > **Hive Read Write Privileges**.

d. In the database list, click the specified database **hdb** to display the **htable** table in the database. Click the **htable** table to display the **hcol** column in the table.

e. In the **Permission** column of the **hcol** column, select **SELECT** and **INSERT**.

f. Click **OK**.

NOTE

In role management, after the metadata permission is granted, the HDFS permission is automatically granted. Therefore, after the column permission is granted, the HDFS ACL permission for all files of the table is automatically granted.

- Granting Database Permissions

  Users have complete permission on the databases created by themselves in Hive and the HDFS. To access the databases created by others, they need to be granted the permission. After the Hive metadata permission is granted, the HDFS permission is automatically granted. The procedure for granting a role the permission of querying data and creating tables in database **hdb** is as follows. Other types of database operation permission are not supported.

  a. On FusionInsight Manager, choose **System** > **Permission** > **Role**.

  b. Locate the row that contains **hrole**, and click **Modify**.

  c. Choose *Name of the desired cluster* > **Hive** > **Hive Read Write Privileges**.

  d. In the **Permission** column of the **hdb** database, select **SELECT** and **CREATE**.

  e. Click **OK**.

  NOTE

  - Any permission for a table in the database is automatically associated with the HDFS permission for the database directory to facilitate permission management. When any permission for a table is canceled, the system does not automatically cancel the HDFS permission for the database directory to ensure performance. In this case, users can only log in to the database and view table names.
  - When the query permission on a database is added to or deleted from a role, the query permission on tables in the database is automatically added to or deleted from the role.
  - If the number of partitions in the database exceeds one million and all partitions are in the table directory in MRS 3.2.0 or later clusters, to accelerate granting permissions to the database, log in to FusionInsight Manager, click **Cluster** and choose **Services** > **Hive** . On the page that is displayed, click **Configurations** and then **All Configurations**. In the navigation pane on the left, choose **MetaStore(Role)** > **Customization**, add the **hive-ext.skip.grant.partition** parameter, and set it to **true**. After this parameter is added, partition scanning is skipped when permissions are granted to the database. You need to restart the MetaStore instance for the modification to take effect.

Concepts

Table 12-4 Scenarios of using Hive tables, columns, or databases

Scenario	Required Permission
DESCRIBE TABLE	SELECT
SHOW PARTITIONS	SELECT
ANALYZE TABLE	SELECT and INSERT
SHOW COLUMNS	SELECT
SHOW TABLE STATUS	SELECT
SHOW TABLE PROPERTIES	SELECT
SELECT	SELECT
EXPLAIN	SELECT
CREATE VIEW	SELECT, Grant Of Select, and CREATE
SHOW CREATE TABLE	SELECT and Grant Of Select
CREATE TABLE	CREATE
ALTER TABLE ADD PARTITION	INSERT
INSERT	INSERT
INSERT OVERWRITE	INSERT and DELETE
LOAD	INSERT and DELETE
ALTER TABLE DROP PARTITION	DELETE
CREATE FUNCTION	Hive Admin Privilege
DROP FUNCTION	Hive Admin Privilege
ALTER DATABASE	Hive Admin Privilege

12.1.4 Granting Hive User Permissions to Use Other Components

Scenario

Hive may need to be associated with other components. For example, Yarn permissions are required in the scenario of using HQL statements to trigger MapReduce jobs, and HBase permissions are required in the Hive over HBase scenario. The following describes the operations in the two scenarios.

NOTE

- In security mode, Yarn and HBase permission management is enabled by default. Therefore, Yarn and HBase permissions need to be configured by default.
- In common mode, Yarn and HBase permission management is disabled by default. That is, any user has permissions. Therefore, YARN and HBase permissions does not need to be configured by default. If a user enables the permission management by modifying the Yarn or HBase configurations, the Yarn and HBase permissions then need to be configured.
- If the current component uses Ranger for permission control, you need to configure permission management policies based on Ranger. For details, see **Adding a Ranger Access Permission Policy for Hive**.

Prerequisites

- The Hive client has been installed. For example, the installation directory is **/opt/client**.
- You have obtained a user account with the administrator permissions, such as **admin**.

Procedure

Association with Yarn

Yarn permissions are required when HQL statements, such as **insert**, **count**, **distinct**, **group by**, **order by**, **sort by**, and **join**, are used to trigger MapReduce jobs. The following uses the procedure for assigning a role the permissions to run the **count** statements in the **thc** table as an example.

Step 1 Create a role on FusionInsight Manager.

Step 2 In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Yarn** > **Scheduler Queue** > **root**.

Step 3 In the **Permission** column of the **default** queue, select **Submit** and click **OK**.

Step 4 In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Hive** > **Hive Read Write Privileges** > **default**. Select **SELECT** for table **thc**, and click **OK**.

----End

Hive over HBase Authorization

After the permissions are assigned, you can use HQL statements that are similar to SQL statements to access HBase tables from Hive. The following uses the procedure for assigning a user the rights to query HBase tables as an example.

Step 1 On the role management page of FusionInsight Manager, create an HBase role, for example, **hive_hbase_create**, and grant the permission to create HBase tables.

In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **HBase** > **HBase Scope** > **global**. Select **Create** of the namespace **default**, and click **OK**.

Step 2 On FusionInsight Manager, create a human-machine user, for example, **hbase_creates_user**, add the user to the **hive** group, and bind the

hive_hbase_create role to the user so that the user can create Hive and HBase tables.

Step 3 If the current component uses Ranger for permission control, grant the create permission for **hive_hbase_create** or **hbase_creates_user**. For details, see **Adding a Ranger Access Permission Policy for Hive**.

Step 4 Log in to the node where the client is installed as the client installation user.

Step 5 Run the following command to configure environment variables:

source /opt/client/bigdata_env

Step 6 Run the following command to authenticate the user:

kinit hbase_creates_user

Step 7 Run the following command to go to the shell environment of the Hive client:

beeline

Step 8 Run the following command to create a table in Hive and HBase, for example, the **thh** table.

CREATE TABLE *thh***(id int, name string, country string) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES("hbase.columns.mapping" = "cf1:id,cf1:name,:key") TBLPROPERTIES ("hbase.table.name" = "***thh***");**

The created Hive table and the HBase table are stored in the Hive database **default** and the HBase namespace **default**, respectively.

Step 9 On the role management page of FusionInsight Manager, create a role, for example, **hive_hbase_select**, and assign the role the permission to query the Hive table **thh** and the HBase table **thh**.

1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **HBase** > **HBase Scope** > **global** > **default**. Select **read** of the **thh** table, and click **OK** to grant the table query permission to the HBase role.
2. Edit the role. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **HBase** > **HBase Scope** > **global** > **hbase**, select **Execute** for **hbase:meta**, and click **OK**.
3. Edit the role. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Hive** > **Hive Read Write Privileges** > **default**. Select **SELECT** for the **thh** table, and click **OK**.

Step 10 On FusionInsight Manager, create a human-machine user, for example, **hbase_select_user**, add the user to the **hive** group, and bind the **hive_hbase_select** role to the user so that the user can query Hive and HBase tables.

Step 11 Run the following command to configure environment variables:

source /opt/client/bigdata_env

Step 12 Run the following command to authenticate users:

kinit hbase_select_user

Step 13 Run the following command to go to the shell environment of the Hive client:

beeline

Step 14 Run the following command to use an HQL statement to query HBase table data:

select * from thh;

----End

12.2 Using the Hive Client

Scenario

This section guides users to use a Hive client in an O&M or service scenario.

Prerequisites

- The client has been installed. For details, see **Installing the Client**. For example, the client is installed in the **/opt/hadoopclient** directory. The client directory in the following operations is only an example. Change it to the actual installation directory.
- Service component users have been created by the MRS cluster administrator. In security mode, machine-machine users need to download the keytab file. A human-machine user must change the password upon the first login.

Using the Hive Client

Step 1 Log in to the node where the client is installed as the client installation user.

Step 2 Run the following command to go to the client installation directory:

cd /opt/hadoopclient

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Log in to the Hive client based on the cluster authentication mode.

- In security mode, run the following command to complete user authentication and log in to the Hive client:

  kinit *Component service user*

  beeline

- In common mode, run the following command to log in to the Hive client. If no component service user is specified, the current OS user is used to log in to the Hive client.

  beeline -n *component service user*

Step 5 Run the following command to execute the HCatalog client command:

hcat -e *"cmd"*

cmd must be a Hive DDL statement, for example, **hcat -e "show tables"**.

NOTE

- To use the HCatalog client, choose **More** > **Download Client** on the service page to download the clients of all services. This restriction does not apply to the beeline client.
- Due to permission model incompatibility, tables created using the HCatalog client cannot be accessed on the HiveServer client. However, the tables can be accessed on the WebHCat client.
- If you use the HCatalog client in Normal mode, the system performs DDL commands using the current user who has logged in to the operating system.
- Exit the beeline client by running the **!q** command instead of by pressing **Ctrl + C**. Otherwise, the temporary files generated by the connection cannot be deleted and a large number of junk files will be generated as a result.
- If multiple statements need to be entered during the use of beeline clients, separate the statements from each other using semicolons (**;**) and set the value of **entireLineAsCommand** to **false**.

  Setting method: If beeline has not been started, run the **beeline --entireLineAsCommand=false** command. If the beeline has been started, run the **!set entireLineAsCommand false** command.

  After the setting, if a statement contains semicolons (**;**) that do not indicate the end of the statement, escape characters must be added, for example, **select concat_ws('\;', collect_set(col1)) from tbl**.

----End

Common Hive Client Commands

The following table lists common Hive Beeline commands.

For more commands, see **https://cwiki.apache.org/confluence/display/Hive/HiveServer2+Clients#HiveServer2Clients-BeelineCommands**.

Table 12-5 Common Hive Beeline commands

Command	Description
set <key>=<value>	Sets the value of a specific configuration variable (key). **NOTE** If the variable name is incorrectly spelled, the Beeline does not display an error.
set	Prints the list of configuration variables overwritten by users or Hive.
set -v	Prints all configuration variables of Hadoop and Hive.
add FILE[S] <filepath> <filepath>* add JAR[S] <filepath> <filepath>* add ARCHIVE[S] <filepath> <filepath>*	Adds one or more files, JAR files, or ARCHIVE files to the resource list of the distributed cache.

Command	Description
add FILE[S] <ivyurl> <ivyurl>* add JAR[S] <ivyurl> <ivyurl>* add ARCHIVE[S] <ivyurl> <ivyurl>*	Adds one or more files, JAR files, or ARCHIVE files to the resource list of the distributed cache using the lvy URL in the **ivy://goup:module:version?query_string** format.
list FILE[S] list JAR[S] list ARCHIVE[S]	Lists the resources that have been added to the distributed cache.
list FILE[S] <filepath>* list JAR[S] <filepath>* list ARCHIVE[S] <filepath>*	Checks whether given resources have been added to the distributed cache.
delete FILE[S] <filepath>* delete JAR[S] <filepath>* delete ARCHIVE[S] <filepath>*	Deletes resources from the distributed cache.
delete FILE[S] <ivyurl> <ivyurl>* delete JAR[S] <ivyurl> <ivyurl>* delete ARCHIVE[S] <ivyurl> <ivyurl>*	Delete the resource added using **<ivyurl>** from the distributed cache.
reload	Enable HiveServer2 to discover the change of the JAR file **hive.reloadable.aux.jars.path** in the specified path. (You do not need to restart HiveServer2.) Change actions include adding, deleting, or updating JAR files.
dfs <dfs command>	Runs the **dfs** command.
<query string>	Executes the Hive query and prints the result to the standard output.

12.3 Using Hive for Data Analysis

Hive is a data warehouse framework built on Hadoop. It maps structured data files to a database table and provides SQL-like functions to analyze and process data. It also allows you to quickly perform simple MapReduce statistics using SQL-like statements without the need of developing a specific MapReduce application. It is suitable for statistical analysis of data warehouses.

Background

Suppose a user develops an application to manage users who use service A in an enterprise. The procedure of operating service A on the Hive client is as follows:

Operations on common tables:

- Create the **user_info** table.
- Add users' educational backgrounds and professional titles to the table.
- Query user names and addresses by user ID.
- Delete the user information table after service A ends.

Table 12-6 User information

ID	Name	Gender	Age	Address
12005000201	A	Male	19	City A
12005000202	B	Female	23	City B
12005000203	C	Male	26	City C
12005000204	D	Male	18	City D
12005000205	E	Female	21	City E
12005000206	F	Male	32	City F
12005000207	G	Female	29	City G
12005000208	H	Female	30	City H
12005000209	I	Male	26	City I
12005000210	J	Female	25	City J

Procedure

Step 1 Log in to the node where the client is installed as the client installation user. For details about how to install the client, see **Installing a Client (MRS 3.x or Later)**.

Step 2 Run the following command to switch to the client directory, for example, **/opt/client**.

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the current user:

kinit *MRS cluster user*

Example: user **kinit hiveuser**

The current user must have the permission to create Hive tables. To create a role with the permission, see **Creating a Role** To bind the role to the current user, see **Creating a User**. If Kerberos authentication is disabled, skip this step.

Step 5 Run the Hive client command to implement service A.

- **Operations on internal tables**
  a. Run the following command to log in to the Hive client CLI:

     beeline

  b. Create the **user_info** user information table according to **Table 12-6** and add data to it.

     create table user_info(id string,name string,gender string,age int,addr string);

     insert into table user_info(id,name,gender,age,addr) values("12005000201","A","Male",19,"City A");

  c. Add users' educational backgrounds and professional titles to the **user_info** table.

     For example, to add educational background and title information about user 12005000201, run the following command:

     alter table user_info add columns(education string,technical string);

  d. Query user names and addresses by user ID.

     For example, to query the name and address of user 12005000201, run the following command:

     select name,addr from user_info where id='12005000201';

  e. Delete the user information table.

     drop table user_info;

  f. Run the following command to exit:

     !q

- **Operations on external partition tables**
  a. You can run the **insert** statement to insert data into an external table or run the **load data** command to import file data from HDFS to an external table. To run the **load data** command to import file data, perform the following operations:

     i. Create a file based on the data in **Table 12-6**. For example, the file name is **txt.log**. Fields are separated by space, and the line feed characters are used as the line breaks.

     ii. Run the following command to upload the file to HDFS.

     hdfs dfs -put txt.log /tmp

  b. Run the following command to create a path for storing external table data.

     hdfs dfs -mkdir /hive/

     hdfs dfs -mkdir /hive/user_info

  c. Run the following command to create a table.

     create external table user_info(id string,name string,gender string,age int,addr string) partitioned by(year string) row format

delimited fields terminated by ' ' lines terminated by '\n' stored as textfile location '/hive/user_info';

NOTE

- **fields terminated**: indicates delimiters, for example, spaces.
- **lines terminated**: indicates line breaks, for example, **\n**.
- **/hive/user_info:** indicates the path of the data file.

d. Import data.

- Execute the insert statement to insert data.

  insert into user_info partition(year="2018") values ("12005000201","A","Male",19,"City A");

- Run the **load data** command to import file data.

  load data inpath '/tmp/txt.log' into table user_info partition (year='2011');

  In the preceding command, **/tmp/txt.log** indicates the data file uploaded to the HDFS in **Step 5.a**.

e. Query the imported data.

select * from user_info;

f. Delete the user information table.

drop table user_info;

g. Run the following command to exit:

!q

----End

12.4 Configuring Hive Data Storage and Encryption

12.4.1 Using HDFS Colocation to Store Hive Tables

Scenario

HDFS Colocation is the data location control function provided by HDFS. The HDFS Colocation API stores associated data or data on which associated operations are performed on the same storage node. Hive supports the HDFS Colocation function. When Hive tables are created, after the locator information is set for table files, data files of related tables are stored on the same storage node when data is inserted into tables using the insert statement (other data import modes are not supported). This ensures convenient and efficient data computing among associated tables. The supported table formats are only TextFile and RCFile.

Procedure

Step 1 Log in to the node where the client is installed as a client installation user.

Step 2 Run the following command to switch to the client installation directory, for example, **/opt/client**:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 If the cluster is in security mode, run the following command to authenticate the user:

kinit *MRS username*

Step 5 Create the *groupid* through the HDFS API.

hdfs colocationadmin -createGroup -groupId *<groupid>* **-locatorIds** *<locatorid1>*,*<locatorid2>*,*<locatorid3>*

NOTE

In the preceding command, *<groupid>* indicates the name of the created group. The group created in this example contains three locators. You can define the number of locators as required.

For details about group ID creation and HDFS Colocation, see HDFS description.

Step 6 Run the following command to log in to the Hive client:

beeline

Step 7 Enable Hive to use colocation.

Assume that **table_name1** and **table_name2** are associated with each other. Run the following statements to create them:

CREATE TABLE *<[db_name.]table_name1>[(col_name data_type , ...)]* **[ROW FORMAT** *<row_format>***] [STORED AS** *<file_format>***] TBLPROPERTIES("groupId"="** *<group>* **","locatorId"="***<locator1>***");**

CREATE TABLE *<[db_name.]table_name2> [(col_name data_type , ...)]* **[ROW FORMAT** *<row_format>***] [STORED AS** *<file_format>***] TBLPROPERTIES("groupId"="** *<group>* **","locatorId"="***<locator1>***");**

After data is inserted into **table_name1** and **table_name2** using the insert statement, data files of **table_name1** and **table_name2** are distributed to the same storage position in the HDFS, facilitating associated operations among the two tables.

----End

12.4.2 Configuring Cold-Hot Separation for Hive Partition Metadata

Cold and Hot Storage of Partitioned Metadata

- The metadata that have not been used for a long time is moved to a backup table to reduce the pressure on metadata databases. This process is called partitioned data freezing. The partitions in which data is moved are cold partitions, partitions that are not frozen are hot partitions. A table with a cold partition is a frozen table. Moving the frozen data back to the original metadata table is called partitioned data unfreezing.
- When a partition is changed from a hot partition to a cold partition, only metadata is identified. The partition path and data file content on the HDFS service side do not change.

Freezing a Partition

The user who creates the table can freeze one or more partitions based on filter criteria. The format is **freeze partitions** *Database name Table name* **where** *Filter criteria*.

Example:

- **freeze partitions testdb.test where year <= 2021;**
- **freeze partitions testdb.test where year<=2021 and month <= 5;**
- **freeze partitions testdb.test where year<=2021 and month <= 5 and day <= 27;**

Unfreezing a Partition

The user who creates the table can unfreeze one or more partitions based on filter criteria. The format is **unfreeze partitions** *Database name Table name* **where** *Filter criteria*. Example:

- **unfreeze partitions testdb.test where year <= 2021;**
- **unfreeze partitions testdb.test where year<=2021 and month <= 5;**
- **unfreeze partitions testdb.test where year<=2021 and month <= 5 and day <= 27;**

Querying Tables with Frozen Data

- Querying all frozen tables in the current database

  show frozen tables;
- Querying all frozen tables in the **dbname** database

  show frozen tables in dbname;

Querying Frozen Partitions of a Frozen Table

Querying frozen partitions

show frozen partitions table;

NOTE

- By default, only partitions of the int, string, varchar, date, or timestamp type can be frozen in the metadata database.
- For external metadata databases, only the Postgres database is supported, and only partitions of the int, string, varchar, or timestamp type can be frozen.
- You need to unfreeze data to restore the metadata of a frozen table using MSCK. If a frozen table has been backed up, you can run **msck repair** to restore the table, and you can only run this command to unfreeze the table.
- You need to unfreeze data before renaming a frozen partition. Otherwise, a message indicating that the partition does not exist is displayed.
- When a table that contains frozen data is deleted, the frozen data is also deleted.
- When a partition that contains frozen data is deleted, information about the frozen partition and HDFS service data is not deleted.
- When you run the **select** command to query data, the criteria for filtering the data in cold partitions is automatically added. The query result does not contain the data in cold partitions.
- When you run the **show partitions table** command to query the partitioned data in the table, the query result does not contain the data in cold partitions. You can run the **show frozen partitions table** command to query frozen partitions.

12.4.3 Hive Supporting ZSTD Compression Formats

Zstandard (ZSTD) is an open-source lossless data compression algorithm. Its compression performance and compression ratio are better than those of other compression algorithms supported by Hadoop. Hive with this feature supports tables in ZSTD compression formats. The ZSTD compression formats supported by Hive include ORC, RCFile, TextFile, JsonFile, Parquet, Squence, and CSV.

You can create a table in ZSTD compression format as follows:

- To create a table in ORC format, specify **TBLPROPERTIES("orc.compress"="zstd")**.

  create table tab_1(...) stored as orc TBLPROPERTIES("orc.compress"="zstd");

- To create a table in Parquet format, specify **TBLPROPERTIES("parquet.compression"="zstd")**.

  create table tab_2(...) stored as parquet TBLPROPERTIES("parquet.compression"="zstd");

- To create a table in other formats or common formats, run the following commands to set the **mapreduce.map.output.compress.codec** and **mapreduce.output.fileoutputformat.compress.codec** parameters to **org.apache.hadoop.io.compress.ZStandardCode**.

  set hive.exec.compress.output=true;

  set mapreduce.map.output.compress=true;

  set mapreduce.map.output.compress.codec=org.apache.hadoop.io.compress.ZStandardCodec;

  set mapreduce.output.fileoutputformat.compress=true;

  set mapreduce.output.fileoutputformat.compress.codec=org.apache.hadoop.io.compress.ZStandardCodec;

set hive.exec.compress.intermediate=true;

create table tab_3(...) stored as textfile;

NOTE

The SQL operations on a table compressed using ZSTD are the same as those on a common compressed table. A table compressed using ZSTD supports addition, deletion, query, and aggregation SQL operations.

12.4.4 Compressing Hive ORC Tables Using ZSTD_JNI

Scenario

ZSTD_JNI is a native implementation of the ZSTD compression algorithm. Compared with ZSTD, ZSTD_JNI has higher compression read/write efficiency and compression ratio, and allows you to specify the compression level as well as the compression mode for data columns in a specific format.

Currently, only ORC tables can be compressed using ASTD_JNI. By contrast, ZSTD enables you to compress tables in the full storage format. Therefore, you are advised to use this feature only when you have high requirements on data compression.

NOTE

This section applies only to MRS 3.2.0 or later.

Example

Step 1 Log in to the node where the client is installed as the Hive client installation user.

Step 2 Run the following command to switch to the client installation directory, for example, **/opt/client**:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Check whether the cluster authentication mode is in security mode.

- If yes, run the following command to perform user authentication and then go to **Step 5**.

  kinit *Hive service user*

- If no, go to **Step 5**.

Step 5 Run the following command to log in to the Hive client:

beeline

Step 6 Create a table in ZSTD_JNI compression format as follows:

- Run the following example command to set the **orc.compress** parameter to **ZSTD_JNI** when using this compression algorithm to create an ORC table:

  create table tab_1(...) stored as orc TBLPROPERTIES("orc.compress"="ZSTD_JNI");

- The compression level of ZSTD_JNI ranges from 1 to 19. A larger value indicates a higher compression ratio but a slower read/write speed. A smaller value indicates a lower compression ratio but a faster compression speed compared with read/write speed and the other way around. The default value is **6**. You can set the compression level through the **orc.global.compress.level** parameter, as shown in the follows.

  create table tab_1(...) stored as orc TBLPROPERTIES("orc.compress"="ZSTD_JNI", 'orc.global.compress.level'='3');

- This compression algorithm allows you to compress service data and columns in a specific data format. Currently, data in the following formats is supported: JSON data columns, Base64 data columns, timestamp data columns, and UUID data columns. You can achieve this function by setting the **orc.column.compress** parameter during table creation.

  The following example code shows how to use ZSTD_JNI to compress data in the JSON, Base64, timestamp, and UUID formats.

  create table test_orc_zstd_jni(f1 int, f2 string, f3 string, f4 string, f5 string) stored as orc

  TBLPROPERTIES('orc.compress'='ZSTD_JNI', 'orc.column.compress'='[{"type":"cjson","columns":"f2"}, {"type":"base64","columns":"f3"},{"type ":"gorilla","columns":{"format": "yyyy-MM-dd HH:mm:ss.SSS", "columns": "f4"}}, {"type":"uuid","columns":"f5"}]');

  You can insert data in the corresponding format based on the site requirements to further compress the data.

----End

12.4.5 Configuring the Hive Column Encryption

Scenario

Hive supports encryption of one or multiple columns in a table. When creating a Hive table, you can specify the column to be encrypted and encryption algorithm. When data is inserted into the table using the insert statement, the related columns are encrypted. Column encryption can be performed in HDFS tables of only the TextFile and SequenceFile file formats. The Hive column encryption does not support views and the Hive over HBase scenario.

Hive supports two column encryption algorithms, which can be specified during table creation:

- AES (the encryption class is org.apache.hadoop.hive.serde2.AESRewriter)
- SMS4 (the encryption class is org.apache.hadoop.hive.serde2.SMS4Rewriter)

NOTE

- In national cryptographic cluster scenarios, Hive column encryption supports only table creation using the SMS4 algorithm.
- When you import data from a common Hive table into a Hive column encryption table, you are advised to delete the original data from the common Hive table as long as doing this does not affect other services. Retaining an unencrypted table poses security risks.

Procedure

Step 1 Specify the column to be encrypted and encryption algorithm when creating a table.

create table*<[db_name.]table_name> (<col_name1> <data_type> ,<col_name2> <data_type>,<col_name3> <data_type>,<col_name4> <data_type>)* ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe' WITH SERDEPROPERTIES ('column.encode.columns'='*<col_name2>,<col_name3>*', 'column.encode.classname'='org.apache.hadoop.hive.serde2.AESRewriter')STORED AS TEXTFILE;

Alternatively, use the following statement:

create table *<[db_name.]table_name> (<col_name1> <data_type> ,<col_name2> <data_type>,<col_name3> <data_type>,<col_name4> <data_type>)* ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe' WITH SERDEPROPERTIES ('column.encode.indices'='1,2', 'column.encode.classname'='org.apache.hadoop.hive.serde2.SMS4Rewriter') STORED AS TEXTFILE;

NOTE

- The numbers used to specify encryption columns start from 0. 0 indicates column 1, 1 indicates column 2, and so on.
- When creating a table with encrypted columns, ensure that the directory where the table resides is empty.

Step 2 Insert data into the table using the insert statement.

Assume that the test table exists and contains data.

insert into table *<table_name>* select *<col_list>* from test;

----End

12.5 Hive on HBase

12.5.1 Configuring Hive on HBase in Across Clusters with Mutual Trust Enabled

For mutually trusted Hive and HBase clusters with Kerberos authentication enabled, you can access the HBase cluster and synchronize its key configurations to HiveServer of the Hive cluster.

Prerequisites

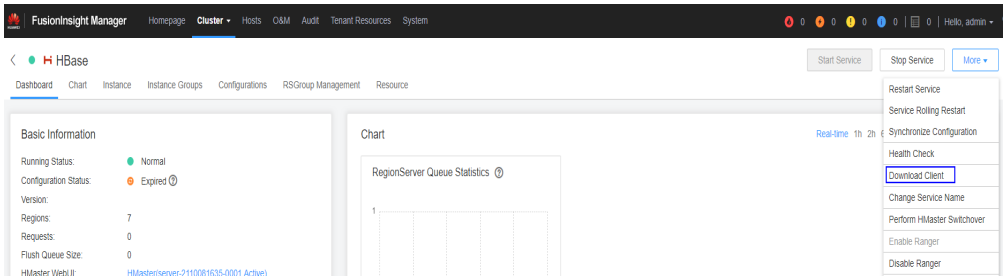
The mutual trust relationship has been configured between the two security clusters with Kerberos authentication enabled.

Procedure for Configuring Hive on HBase Across Clusters

Step 1 Download the HBase configuration file and decompress it.

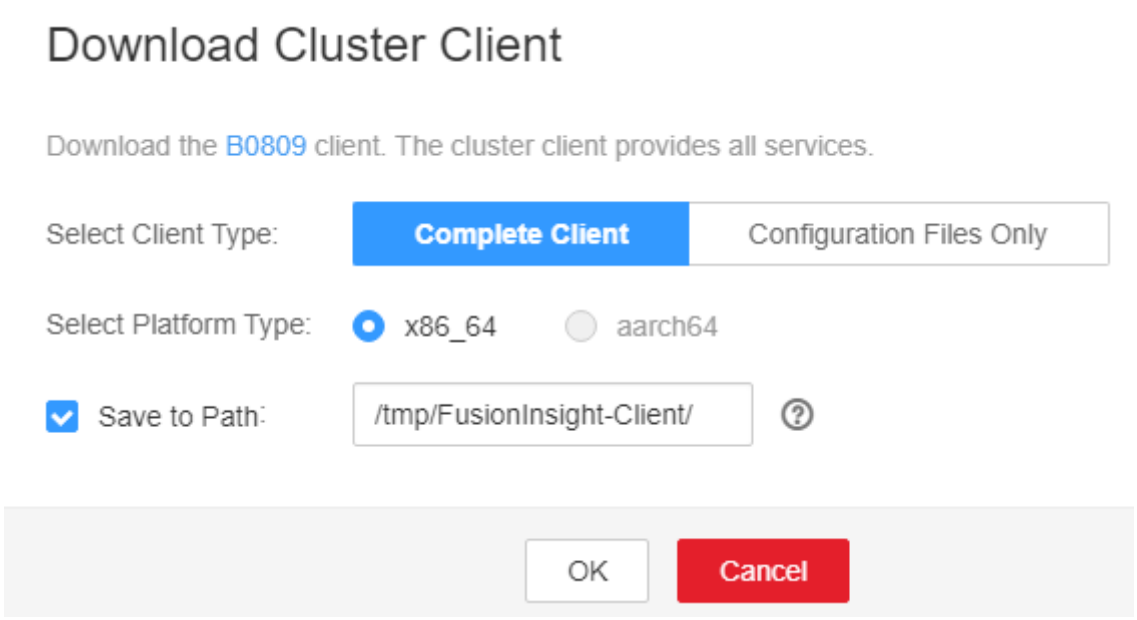
1. Log in to FusionInsight Manager of the target HBase cluster, click **Cluster** and choose **Services** > **HBase**.
2. Choose **More** > **Download Client**.

Figure 12-1 Downloading the HBase client



3. Download the HBase configuration file and choose **Configuration Files only** for **Select Client Type**.

Figure 12-2 Downloading the HBase configuration file



Step 2 Log in to FusionInsight Manager of the source Hive cluster.

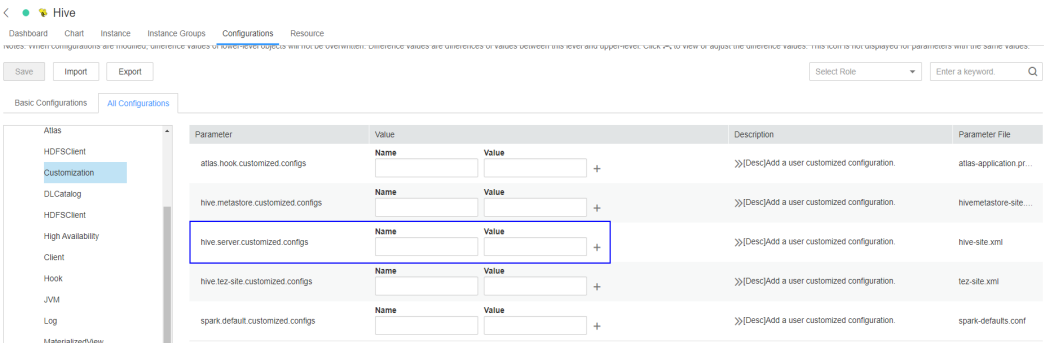
Step 3 Click **Cluster**, choose **Services** > **Hive**, click **Configurations** and then **All Configurations**. On the displayed page, add the following parameters to the **hive-site.xml** configuration file of the HiveServer role.

Search for the following parameters in the **hbase-site.xml** configuration file of the downloaded HBase client and add them to HiveServer:

- hbase.security.authentication
- hbase.security.authorization

- hbase.zookeeper.property.clientPort
- hbase.zookeeper.quorum (The domain name needs to be converted into an IP address.)
- hbase.regionserver.kerberos.principal
- hbase.master.kerberos.principal

Figure 12-3 Custom configurations of the HiveServer role



Step 4 Save the configurations and restart Hive.

----End

12.5.2 Deleting Single-Row Records from Hive on HBase

Scenario

Due to the limitations of underlying storage systems, Hive does not support the ability to delete a single piece of table data. In Hive on HBase, MRS Hive supports the ability to delete a single piece of HBase table data. Using a specific syntax, Hive can delete one or more pieces of data from an HBase table.

Table 12-7 Permissions required for deleting single-row records from the Hive on HBase table

Cluster Authentication Mode	Required Permission
Security mode	SELECT, INSERT, and DELETE
Common mode	None

Procedure

Step 1 To delete some data from an HBase table, run the following HQL statement:

remove table ***<table_name>*** **where** ***<expression>*****;**

In the preceding information, *<expression>* specifies the filter condition of the data to be deleted. *<table_name>* indicates the Hive on HBase table from which data is to be deleted.

----End

12.6 Using Hive to Read Data in a Relational Database

Scenario

Hive allows users to create external tables to associate with other relational databases. External tables read data from associated relational databases and support Join operations with other tables in Hive.

Currently, Hive can be used to read data from DB2 and Oracle relational databases.

Prerequisites

The Hive client has been installed.

Procedure

Step 1 Log in to the node where the Hive client is installed as the Hive client installation user.

Step 2 Run the following command to go to the client installation directory:

cd *Client installation directory*

For example, if the client installation directory is **/opt/client**, run the following command:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Check whether the cluster authentication mode is Security.

- If yes, run the following command to authenticate the user:

  kinit *Hive service user*
- If no, go to **Step 5**.

Step 5 Run the following command to upload the driver JAR package of the relational database to be associated to an HDFS directory.

hdfs dfs -put *directory where the JAR package is located HDFS directory to which the JAR is uploaded*

For example, to upload the Oracle driver JAR package in **/opt** to the **/tmp** directory in HDFS, run the following command:

hdfs dfs -put /opt/ojdbc6.jar /tmp

Step 6 Create an external table on the Hive client to associate with the relational database, as shown in the following example.

```
-- Example of associating with an Oracle Linux 6 database
-- In security mode, set the admin permission.
set role admin;
-- Upload the driver JAR package of the relational database to be associated. The driver JAR packages vary
```



```
according to databases.
ADD JAR hdfs:///tmp/ojdbc6.jar;

CREATE EXTERNAL TABLE ora_test
-- The Hive table must have one more column than the database return result. This column is used for paging query.
(id STRING,rownum string)
STORED BY 'com.qubitproducts.hive.storage.jdbc.JdbcStorageHandler'
TBLPROPERTIES (
-- Relational database table type
"qubit.sql.database.type" = "ORACLE",
-- Connect to the URL of the relational database through JDBC. (The URL formats vary according to databases.)
"qubit.sql.jdbc.url" = "jdbc:oracle:thin:@//10.163.0.1:1521/mydb",
-- Relational database driver class type
"qubit.sql.jdbc.driver" = "oracle.jdbc.OracleDriver",
-- SQL statement queried in the relational database. The result is returned to the Hive table.
"qubit.sql.query" = "select name from aaa",
-- (Optional) Match the Hive table columns to the relational database table columns.
"qubit.sql.column.mapping" = "id=name",
-- Relational database user
"qubit.sql.dbcp.username" = "test",
-- Relational database password. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.
"qubit.sql.dbcp.password" = "xxx");
```

NOTE

- If the security mode is used, the user who creates the table must have the **ADMIN** permission.
- The **ADD JAR** path is subject to the actual path.

----End

12.7 Hive Supporting Reading Hudi Tables

Hive External Tables Corresponding to Hudi Tables

A Hudi source table corresponds to a copy of HDFS data. The Hudi table data can be mapped to a Hive external table through the Spark component, Flink component, or Hudi client. Based on the external table, Hive can easily perform real-time query, read-optimized view query, and incremental view query.

NOTE

- Different view queries are provided for different types of Hudi source tables:
  - When the Hudi source table is a Copy-On-Write (COW) table, it can be mapped to a Hive external table. The table supports real-time query and incremental view query.
  - When the Hudi source table is a Merge-On-Read (MOR) table, it can be mapped to two Hive external tables (RO table and RT table). The RO table supports read-optimized view query, and the RT table supports real-time view query and incremental view query.
- Hive external tables cannot be added, deleted, or modified (including **insert**, **update**, **delete**, **load**, **merge**, **alter** and **msck**). Only the query operation (**select**) is supported.
- Granting table permissions: The update, alter, write, and all permissions cannot be modified.
- Backup and restoration: The RO and RT tables are mapped from the same Hudi source table. When one table is backed up, the other table is also backed up. The same applies to restoration. Therefore, only one table needs to be backed up.
- Component versions:
  - Hive: FusionInsight_HD_xxx; Hive kernel version 3.1.0.
  - Spark2x: FusionInsight_Spark2x_xxx; Hudi kernel version 0.11.0

Creating Hive External Tables Corresponding to Hudi Tables

Generally, Hudi table data is synchronized to Hive external tables when the data is imported to the lake. In this case, you can directly query the corresponding Hive external tables in Beeline. If the data is not synchronized to the Hive external tables, you can use the Hudi client tool **run_hive_sync_tool.sh** to synchronize data manually.

Querying Hive External Tables Corresponding to Hudi Tables

Prerequisites

Before using Hive to perform incremental query on Hudi tables, you need to set another three parameters in **Table 12-8**. The three parameters are table-level parameters. Each Hudi source table corresponds to three parameters, where **hudisourcetablename** indicates the name of the Hudi source table (not the name of the Hive external table).

Table 12-8 Parameter description

Parameter	Default Value	Description
hoodie. hudisourcetable-name.consume.mode	None	Query mode of the Hudi table. • Incremental query: Set it to **INCREMENTAL**. • Non-incremental query: Do not set this parameter or set it to **SNAPSHOT**.

Parameter	Default Value	Description
hoodie. hudisourcetable-name.consume.start.timestamp	None	Start time of the incremental query on the Hudi table. ● Incremental query: start time of the incremental query. ● Non-incremental query: Do not set this parameter.
hoodie. hudisourcetable-name.consume.max.commits	None	The incremental query on the Hudi table is based on the number of commits after **hoodie.hudisourcetable-name.consume.start.timestamp**. ● Incremental query: number of commits. For example, if this parameter is set to **3**, data after three commits from the specified start time is queried. If this parameter is set to **-1**, all data committed after the specified start time is queried. ● Non-incremental query: Do not set this parameter.

Querying a Hudi COW Table

For example, the name of a Hudi source table of the COW type is **hudicow**, and the name of the mapped Hive external table is **hudicow**.

- Real-time view query on the COW table:

  Select * from hudicow;

- Incremental query on the COW table: Set three incremental query parameters based on the name of the Hudi source table. The **where** clause of the incremental query statements must contain **`_hoodie_commit_time`>'***xxx*'**, where *xxx* indicates the value of **hoodie.hudisourcetablename.consume.start.timestamp**.

  set hoodie.hudicow.consume.mode= INCREMENTAL;

  set hoodie.hudicow.consume.max.commits=3;

  set hoodie.hudicow.consume.start.timestamp= 20200427114546;

  select count(*) from hudicow where `_hoodie_commit_time`>'20200427114546';

Querying a Hudi MOR Table

For example, the name of a Hudi source table of the MOR type is **hudimor**, and the two mapped Hive external tables are **hudimor_ro** (RO table) and **hudimor_rt** (RT table).

- Read-optimized view query on the RO table:

  Select * from hudicow_ro;

- Real-time view query on the RT table:

  Select * from hudicow_rt;

- Incremental query on the RT table: Set three incremental query parameters based on the name of the Hudi source table. The **where** clause of the incremental query statements must contain **`_hoodie_commit_time`>'***xxx*'**, where *xxx* indicates the value of **hoodie.hudisourcetablename.consume.start.timestamp**.

  set hoodie.hudimor.consume.mode=INCREMENTAL;

  set hoodie.hudimor.consume.max.commits=-1;

  set hoodie.hudimor.consume.start.timestamp=20210207144611;

  select * from hudimor_rt where `_hoodie_commit_time`>'20210207144611';

  NOTE

  set hoodie.hudisourcetablename.consume.mode=INCREMENTAL; is used only for the incremental query on the table. To switch to another query mode, run **set hoodie.hudisourcetablename.consume.mode=SNAPSHOT;**.

Querying Hive External Tables Corresponding to Hudi Schema Evolution Tables

If the Hudi table is a schema evolution table (some fields in the table have been modified), you need to set **set hive.exec.schema.evolution** to **true** when Hive queries the table.

The following uses the query of the real-time view of a COW table as an example. To query other views, you need to add this parameter.

- Real-time view query on the COW table:

  set hive.exec.schema.evolution=true;

  select * from *hudicow*;

12.8 Enterprise-Class Enhancements of Hive

12.8.1 Storing Hive Table Partitions to OBS and HDFS

Scenario

In the scenario where storage and compute resources are separated, you can specify different storage sources, for example, OBS or HDFS, for partitions in a Hive partitioned table.

NOTE

This feature applies only to MRS 3.2.0 or later. This section describes the capability of specifying storage sources for partitioned tables. For details about how to connect Hive to OBS in the storage-compute decoupling scenario, see **Interconnecting Hive with OBS**.

Prerequisites

The Hive client has been installed.

Example

1. Log in to the node where the Hive client is installed as the Hive client installation users.
2. Run the following command to go to the client installation directory:

   cd *Client installation directory*

   For example, if the client installation directory is **/opt/client**, run the following command:

   cd /opt/client
3. Run the following command to configure environment variables:

   source bigdata_env
4. Check whether the cluster authentication mode is in security mode.
   - If yes, run the following command to authenticate the user:

     kinit *Hive service user*
   - If no, go to **5**.
5. Run the following command to log in to the Hive client:

   beeline
6. Run the following commands to create a Hive partitioned table named **table_1**, and set the path of partitions **pt=?2021-12-12** and **pt='2021-12-18** to **hdfs//***xxx* and **obs://***xxx* respectively:

   create table table_1(id string) partitioned by(pt string) [stored as [orc|textfile|parquet|...]];

   alter table table_1 add partition(pt='2021-12-12') location 'hdfs://*xxx***';**

   alter table table_1 add partition(pt='2021-12-18') location 'obs://*xxx*';
7. After data is inserted into **table_1**, it is stored in the corresponding storage source. You can run the **desc** command to view the location of each partition.

   desc formatted table_1 partition(pt='2021-12-18');

12.8.2 Configuring Automatic Removal of Old Data in the Hive Directory to the Recycle Bin

After this function is enabled, run the following command to write a directory into Hive: **insert overwrite directory "***/path1***" ...**. After the operation is successfully performed, the old data is removed to the recycle bin, and the directory cannot be an existing database path in the Hive metastore.

Procedure

Step 1 Log in to FusionInsight Manager. Click **Cluster** > **Services** > **Hive**, click **Configurations** > **All Configurations**.

Step 2 Choose **HiveServer(Role)** > **Customization**, add a customized parameter to the **hive-site.xml** parameter file, set **Name** to **hive.overwrite.directory.move.trash**, and set **Value** to **true**.

Step 3 Click **Save** to save the configuration. Click **Instances**, select all Hive instances, choose **More** > **Restart Instance**, enter the password of the current user, and click **OK** to restart all Hive instances.

----End

12.8.3 Configuring Hive to Insert Data to a Directory That Does Not Exist

With this function enabled, run the **insert overwrite directory** */path1/path2/path3*... command to write a subdirectory. The permission of the */path1/path2* directory is 700, and the owner is the current user. If the */path3* directory does not exist, it is automatically created and data is written successfully.

NOTE

This function is supported when **hive.server2.enable.doAs** is set to **true** in earlier versions. This version supports the function when **hive.server2.enable.doAs** is set to **false**.

Procedure

Step 1 Log in to FusionInsight Manager. Click **Cluster** > **Services** > **Hive**, click **Configurations** > **All Configurations**.

Step 2 Choose **HiveServer(Role)** > **Customization**, add a customized parameter to the **hive-site.xml** parameter file, set **Name** to **hive.overwrite.directory.move.trash**, and set **Value** to **true**.

Step 3 Click **Save** to save the configuration. Click **Instance**, select all Hive instances, choose **More** > **Restart Instance**, enter the password of the current user, and click **OK** to restart all Hive instances.

----End

12.8.4 Forbidding Location Specification When Hive Internal Tables Are Created

The **hive.internaltable.notallowlocation** parameter can be used to prevent Location from being specified during Hive internal table creation. That is, after the table is created successfully, the location path of the table is created in the current default warehouse directory and cannot be specified in other directories. If Location is specified when creating an internal table, the creation fails.

NOTE

After this function is enabled, the location keyword cannot be specified during the creation of a Hive internal table. The table creation statement is restricted. If a table that has been created in the database is not stored in the default directory **/warehouse**, the **location** keyword can still be specified when the database creation, table script migration, or metadata recreation operation is performed by disabling this function temporarily.

Procedure

Step 1 Log in to FusionInsight Manager. Click **Cluster** > **Services** > **Hive**, click **Configurations** > **All Configurations**.

Step 2 Choose **HiveServer(Role)** > **Customization**, add a customized parameter to the **hive-site.xml** parameter file, set **Name** to **hive.internaltable.notallowlocation**, and set **Value** to **true**.

Step 3 Click **Save** to save the configuration. Click **Instance**, select all Hive instances, choose **More** > **Restart Instance**, enter the password of the current user, and click **OK** to restart all Hive instances.

Step 4 Determine whether to enable this function on the Spark/Spark2x client.

- If yes, download and install the Spark/Spark2x client again.
- If no, no further action is required.

----End

12.8.5 Creating a Foreign Table in a Directory (Read and Execute Permission Granted)

The **hive.restrict.create.grant.external.table** parameter is used to allow users and user groups with the read and execute permissions to create Hive external tables without checking whether the user is the owner of the directory. In addition, the Location directory of the foreign table cannot be in the default **\warehouse** directory. In addition, when authorizing foreign tables, do not change the permission corresponding to the Location directory.

NOTE

After this function is enabled, the function of the foreign table changes greatly. Based on the actual application scenario, determine whether to enable this function.

Procedure

Step 1 Log in to FusionInsight Manager. Click **Cluster**, choose **Services** > **Hive**, click **Configurations**, and then **All Configurations**.

Step 2 Choose **HiveServer(Role)** > **Customization**, add a customized parameter to the **hive-site.xml** parameter file, set **Name** to **hive.restrict.create.grant.external.table**, and set **Value** to **true**.

Step 3 Choose **MetaStore(Role)** > **Customization**, add a customized parameter to the **hivemetastore-site.xml** parameter file, set **Name** to **hive.restrict.create.grant.external.table**, and set **Value** to **true**.

Step 4 Click **Save** to save the configuration. Click **Instances**, select all Hive instances, choose **More** > **Restart Instance**, enter the password of the current user, and click **OK** to restart all Hive instances.

Step 5 Determine whether to enable this function on the Spark/Spark2x client.

- If yes, download and install the Spark/Spark2x client again.
- If no, no further action is required.

----End

12.8.6 Configuring HTTPS/HTTP-based REST APIs

Scenario

WebHCat provides external REST APIs for Hive. By default, the open-source community version uses the HTTP protocol.

MRS Hive supports the HTTPS protocol that is more secure, and enables switchover between the HTTP protocol and the HTTPS protocol.

NOTE

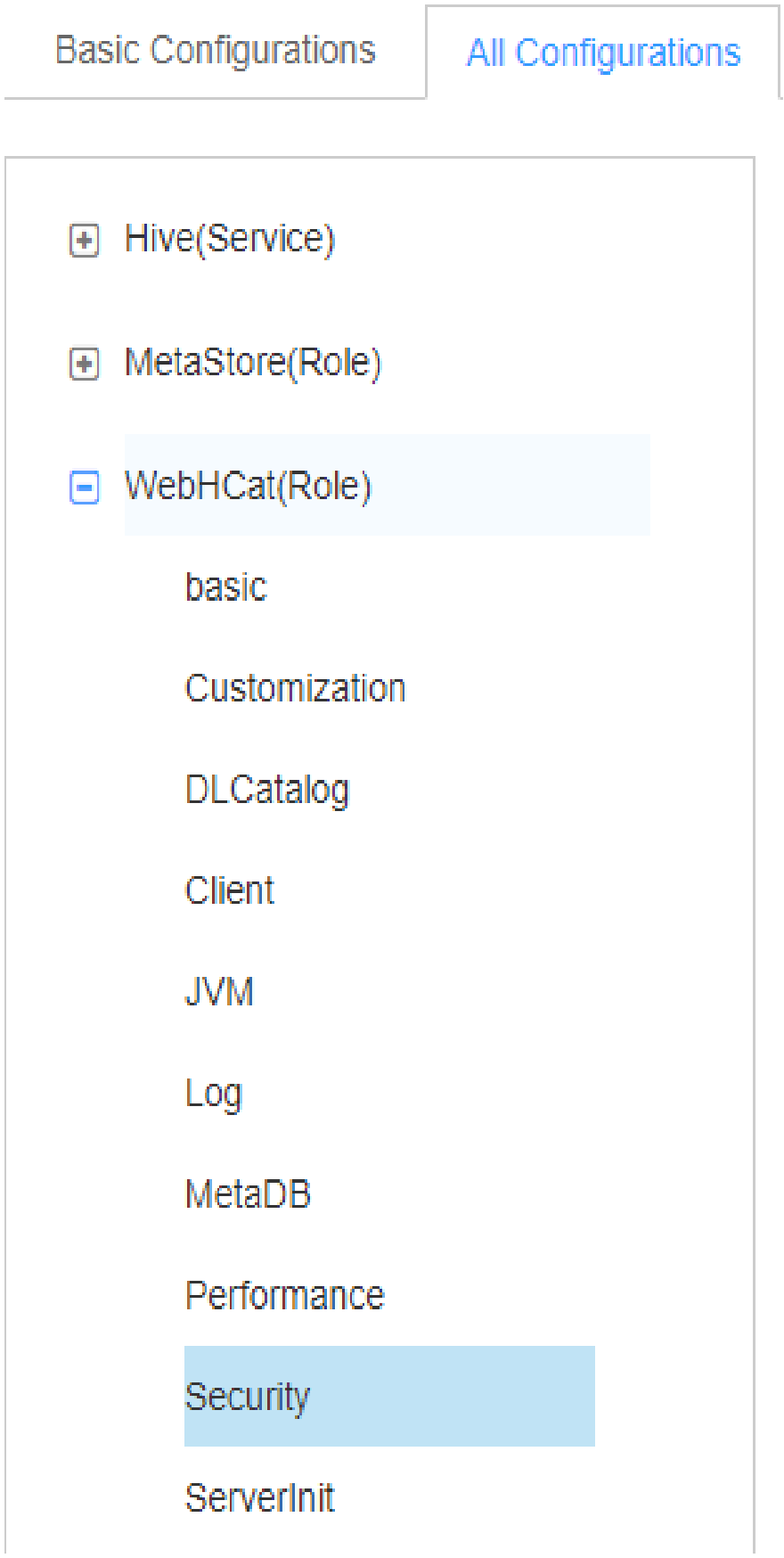
The security mode supports HTTPS and HTTP, and the common mode supports only HTTP.

Procedure

Step 1 Log in to FusionInsight Manager. And choose **Cluster** > **Services** > **Hive** > **Configurations** > **All Configurations**.

Step 2 Modify the Hive configuration.

Choose **WebHCat** > **Security**. On the page that is displayed, select **HTTPS** or **HTTP**. After the modification, restart the Hive service to use the corresponding protocol.



----End

12.8.7 Configuring Hive Transform

Scenario

The Transform function is not allowed by Hive of the open source version.

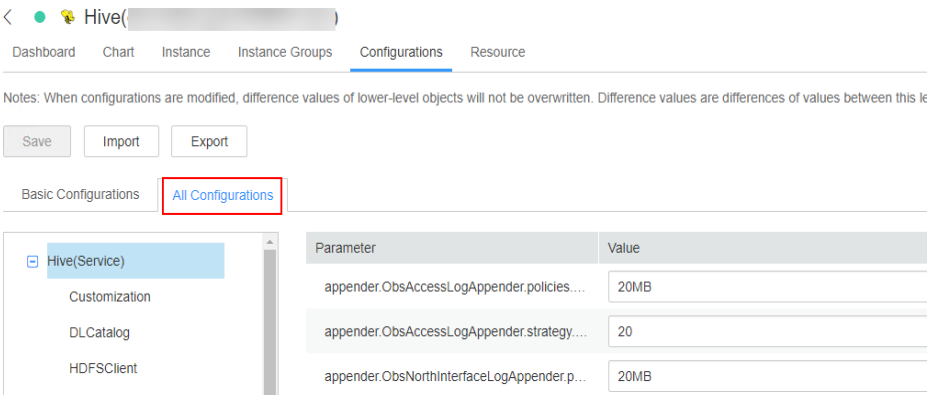
MRS Hive supports the configuration of the Transform function. The function is disabled by default, which is the same as that of the open-source community version.Users can modify configurations of the Transform function to enable the function. However, security risks exist when the Transform function is enabled.

NOTE

The Transform function can be disabled only in security mode.

Procedure

Step 1 Log in to FusionInsight Manager. And choose **Cluster** > **Services** > **Hive** > **Configurations** > **All Configurations**.



Step 2 Enter the parameter name in the search box, search for **hive.security.transform.disallow**, change the parameter value to **true** or **false**, and restart all HiveServer instances.

NOTE

- If this parameter is set to **true**, the Transform function is disabled, which is the same as that in the open-source community version.
- If this parameter is set to **false**, the Transform function is enabled, which poses security risks.

----End

12.8.8 Switching the Hive Execution Engine to Tez

Scenario

Hive can use the Tez engine to process data computing tasks. Before executing a task, you can manually switch the execution engine to Tez.

Prerequisites

The TimelineServer role of the Yarn service has been installed in the cluster and is running properly.

Switching the Execution Engine on the Client to Tez

Step 1 Install and log in to the Hive client. For details, see **Using the Hive Client**.

Step 2 Run the following commands for MRS 3.1.2 to switch the engine and enable the **yarn.timeline-service.enabled** parameter:

set hive.execution.engine=tez;

set yarn.timeline-service.enabled=true;

NOTE

- After **yarn.timeline-service.enabled** is enabled, you can view the details about the tasks executed by the Tez engine on TezUI. After this function is enabled, task information will be reported to TimelineServer. If the TimelineServer instance is faulty, the task will fail.
- Tez uses the ApplicationMaster buffer pool. Therefore, **yarn.timeline-service.enabled** must be enabled before Tez tasks are submitted. Otherwise, this parameter cannot take effect and you need to log in to the client again to configure it.
- When the execution engine needs to be switched to another engine, you need to run the **set yarn.timeline-service.enabled=false** command on the client to disable the **yarn.timeline-service.enabled** parameter.
- To specify a Yarn running queue, run the **set tez.queue.name=default** command on the client.

Step 3 For MRS 3.2.0 and later versions, run the following command to switch the engine:

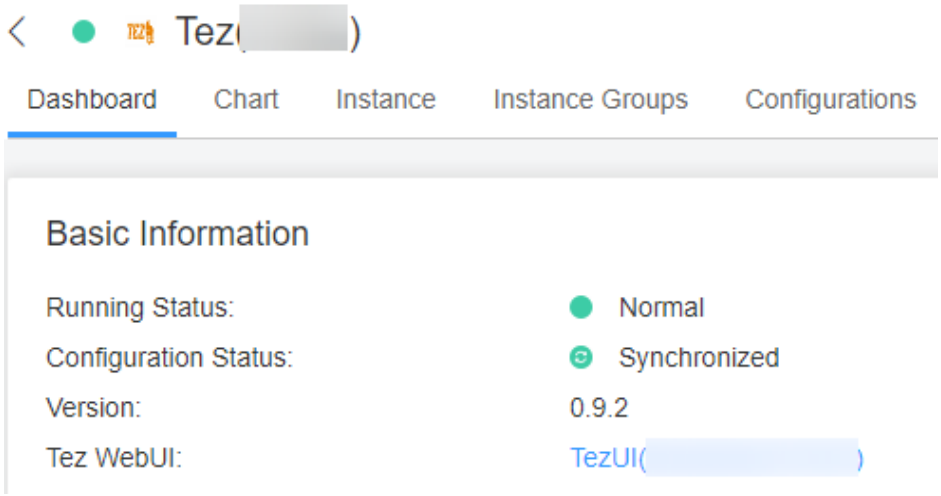
set hive.execution.engine=tez;

NOTE

To specify a running Yarn queue, run the **set tez.queue.name=default** command on the client.

Step 4 Submit and execute the Tez tasks.

Step 5 Log in to FusionInsight Manager. Choose **Cluster** > **Services** > **Tez** > **TezUI** *(host name)* to view the task execution status on the TezUI page.



----End

Switching the Default Execution Engine of Hive to Tez

Step 1 Log in to FusionInsight Manager. Choose **Cluster** > **Services** > **Hive** > **Configurations** > **All Configurations** > **HiveServer(Role)**, and search for **hive.execution.engine**.

Step 2 Set **hive.execution.engine** to **tez**.

Step 3 Choose **Hive(Service)** > **Customization** and search for **yarn.site.customized.configs** for MRS 3.1.2.

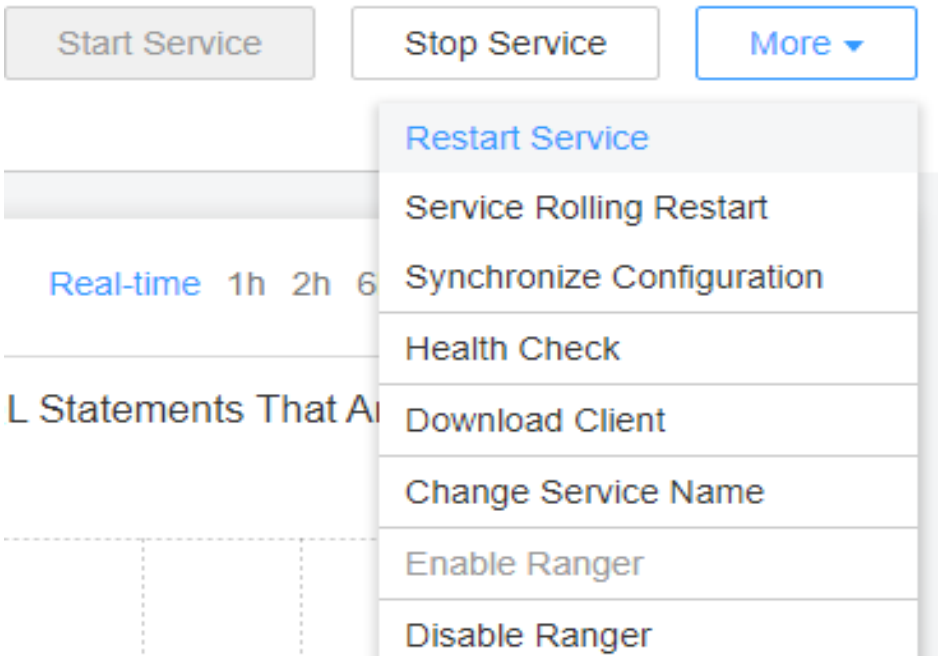
Step 4 Add custom parameter **yarn.timeline-service.enabled** to **yarn.site.customized.configs** and set it to **true** for MRS 3.1.2.

NOTE

- After **yarn.timeline-service.enabled** is enabled, you can view the details about the tasks executed by the Tez engine on TezUI. After this function is enabled, task information will be reported to TimelineServer. If the TimelineServer instance is faulty, the task will fail.
- Tez uses the ApplicationMaster buffer pool. Therefore, **yarn.timeline-service.enabled** must be enabled before Tez tasks are submitted. Otherwise, this parameter cannot take effect and you need to log in to the client again to configure it.
- When the execution engine needs to be switched to another one, you need to set the value of parameter **yarn.timeline-service.enabled** to **false**.

Step 5 Click **Save**. In the displayed confirmation dialog box, click **OK**.

Step 6 Choose **Dashboard** > **More** > **Restart Service**, enter the password of the current user, and click OK to restart the Hive service.



Step 7 Install and log in to the Hive client. For details, see **Using the Hive Client**.

Step 8 Submit and execute the Tez tasks.

Step 9 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Tez** > **TezUI** *(host name)*. On the displayed TezUI page, view the task execution status.

----End

12.8.9 Hive Load Balancing

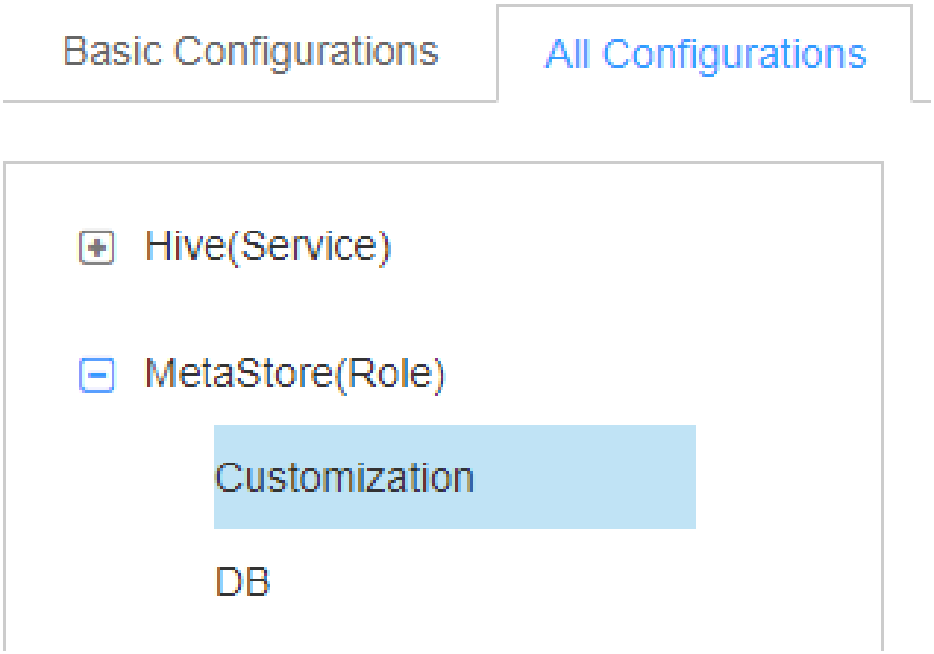
12.8.9.1 Configuring the Maximum Number of Maps for Hive Tasks

The **hive.mapreduce.per.task.max.splits** parameter can be used to limit the maximum number of maps of Hive tasks on the server to prevent performance problems caused by HiveSever overload.

Procedure

Step 1 Log in to FusionInsight Manager. Click **Cluster**, choose **Services** > **Hive**, click **Configurations**, and then **All Configurations**.

Step 2 Choose **MetaStore(Role)** > **Customization**, add a customized parameter to the **hivemetastore-site.xml** parameter file, set **Name** to **hive.mapreduce.per.task.max.splits**, and set the parameter to a large value.



Step 3 Click **Save** to save the configuration. Click **Instances**, select all Hive instances, choose **More** > **Restart Instance**, enter the password of the current user, and click **OK** to restart all Hive instances.

----End

12.8.9.2 Configuring User Lease Isolation to Access HiveServer on a Specified Node

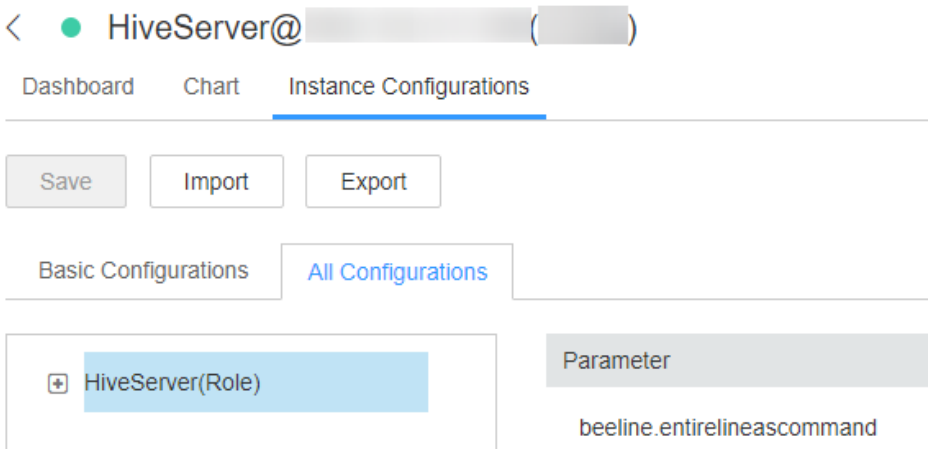
Hive user lease isolation allows specified users to access the HiveServer service on specified nodes, implementing resource isolation for users to access the HiveServer service.

Procedure

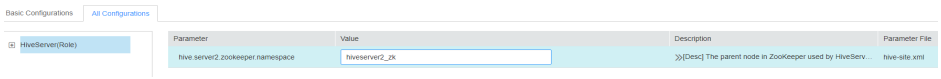
This section describes how to set lease isolation for user **hiveuser** for existing HiveServer instances.

Step 1 Log in to FusionInsight Manager. And Choose **Cluster** > **Services** > **Hive** > **Instances**.

Step 2 In the Instances list, select the HiveServer for which lease isolation is configured and choose **HiveServer** > **Instance Configurations** > **All Configurations**.



Step 3 In the upper right corner of the **All Configurations** page, search for **hive.server2.zookeeper.namespace** and specify its value, for example, **hiveserver2_zk**.



Step 4 Click **Save**. In the dialog box that is displayed, click **OK**.

Step 5 Return to the Instances page, click **Dashboard**, choose **More** > **Restart Service**, enter the password of the current user, and click **OK** to restart the service.

Step 6 Log in to the node where the Hive client is installed as the client installation user and run the following command:

cd *client installation directory*

source bigdata_env

kinit *Hive service user* (Skip this step if Kerberos authentication is not enabled for the cluster.)

Step 7 Run the **beeline -u** command to log in to the client and run the following command:

beeline -u "jdbc:hive2://*IP address of any ZooKeeper instance***:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=***hiveserver2_zk***;sasl.qop=auth-conf;auth=KERBEROS;principal=hive/hadoop.***<System domain name>***@***<System domain name>***"**

- To view the IP address of the ZooKeeper instance, log in to FusionInsight Manager and choose **Cluster** > **Services** > **ZooKeeper** > **Instances**.
- **hiveserver2_zk** is the value of **hive.server2.zookeeper.namespace** in **Step 3**.
- To obtain the system domain name, log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and check the **Local Domain** parameter.
- **hive/hadoop.***<System domain name>* indicates the user name. All letters in the system domain name contained in the user name are in lowercase.

As a result, only the HiveServer whose lease isolation is configured can be logged in.

NOTE

- After this function is enabled, you must run the preceding command during login to access the HiveServer for which lease isolation is configured. If you run the **beeline** command to log in to the client, only the HiveServer that is not isolated by the lease is accessed.
- You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and view the value of **Local Domain**, which is the current system domain name. **hive/hadoop.**<*system domain name*> is the username. All letters in the system domain name contained in the username are lowercase letters.

----End

12.8.9.3 Configuring Component Isolation to Access Hive MetaStore

Scenario

This function restricts components in a cluster to connect to specified Hive Metastore instances. By default, components can connect to all Metastore instances. This function applies only to clusters whose version is MRS 3.2.0 or later.

Currently, only HetuEngine, Hive, Loader, Metadata, Spark2x and Flink can connect to Metastore in a cluster. The Metastore instances can be allocated in a unified manner.

NOTE

- This function only limits the Metastore instances accessed by component servers. Metadata is not isolated.
- Currently, Flink tasks can only connect to Metastore instances through the client.
- When spark-sql is used to execute tasks, the client is directly connected to Metastore. The client needs to be updated for the isolation to take effect.
- This function supports only isolation in the same cluster. If HetuEngine is deployed in different clusters, unified isolation configuration is not supported. You need to modify the HetuEngine configuration to connect to the specified Metastore instance.
- You are advised to configure at least two Metastore instances for each component to ensure availability during isolation configuration.

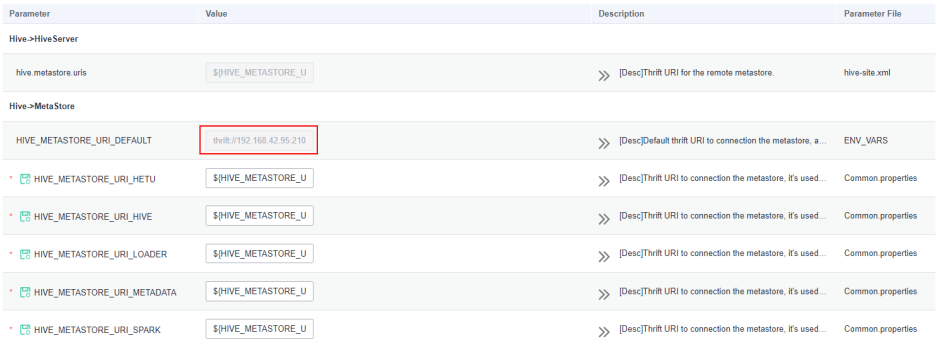
Prerequisites

The Hive service has been installed in the cluster and is running properly.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Hive**. On the displayed page, click the **Configurations** tab and then **All Configurations**, and search for the **HIVE_METASTORE_URI** parameter.

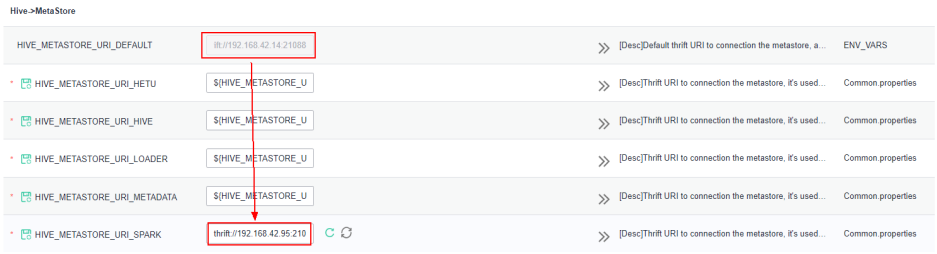
Step 2 Set the value of **HIVE_METASTORE_URI_DEFAULT** to the URI connection string of all Metastore instances.



Step 3 Connect a component to a specified Metastore instance. Copy the value in **Step 2**, modify the configuration items based on the component name, save the modification, and restart the component.

The following example shows how Spark2x connects to only two Metastore instances of Hive.

1. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Hive**. On the displayed page, click the **Configurations** tab and then **All Configurations**, and search for the **HIVE_METASTORE_URI** parameter.
2. Copy the default configuration of **HIVE_METASTORE_URI_DEFAULT** to the URI configuration item of Spark2x. If Spark2x needs to connect to only two Metastore instances, retain two nodes as required. Click **Save**.



3. Choose **Cluster** > **Services** > **Spark2x**. On the displayed page, click the **Instance** tab, select the instances whose configuration has expired, and choose **More** > **Restart Instance**. In the dialog box that is displayed, enter the password and click **OK** to restart the instances.

----End

12.8.9.4 Configuring Load Balancing for HiveMetaStore Client Connections

Scenario

The client connection of Hive MetaStore supports load balancing. That is, heavy load of a single MetaStore node during heavy service traffic can be avoided by connecting to the node with the least connections based on the connection number recorded in ZooKeeper. Enabling this function does not affect the original connection mode.

NOTE

This section applies to MRS 3.2.0 or later.

Procedure

Step 1 Log in to FusionInsight Manager, click **Cluster**, choose **Services** > **Hive**, click **Configurations**, and then **All Configurations**.

Step 2 Search for the **hive.metastore-ext.balance.connection.enable** parameter and set its value to **true**.

Step 3 Click **Save**.

Step 4 Click **Instance**, select all instances, choose **More** > **Restart Instance**, enter the password, and click **OK** to restart all Hive instances.

Step 5 For other components that connect to MetaStore, add the **hive.metastore-ext.balance.connection.enable** parameter and set its value to **true**.

The following example shows how to add this parameter if Spark needs to be connected to MetaStore:

1. Log in to FusionInsight Manager, click **Cluster**, choose **Services** > **Spark**, and click **Configurations**.
2. Click **Customization**, add a custom parameter **hive.metastore-ext.balance.connection.enable** to all **hive-site.xml** parameter files, set its value to **true**, and click **Save**.
3. Click **Instance**, select all configuration-expired instances, choose **More** > **Restart Instance**, enter the password, and click **OK** to restart them.

----End

12.8.10 Configuring Access Control Permission for the Dynamic View of a Hive Single Table

Scenario

This section describes how to create a view on Hive when MRS is configured in security mode, authorize access permissions to different users, and specify that different users access different data.

In the view, Hive can obtain the built-in function **current_user()** of the users who submit tasks on the client and filter the users. This way, authorized users can only access specific data in the view.

NOTE

In normal mode, the **current_user()** function cannot distinguish users who submit tasks on the client. Therefore, the access control function takes effect only for Hive in security mode.

If the **current_user()** function is used in the actual service logic, the possible risks must be fully evaluated during the conversion between the security mode and normal mode.

Operation Example

Step 1 Log in to the node where the Hive client is installed as the Hive client installation user.

Step 2 Run the following commands to switch to the client installation directory, configure environment variables, and authenticate users:

cd *client installation directory*

source bigdata_env

kinit *Hive service user*

Step 3 Run the following command to log in to the Hive client:

beeline

Step 4 The following is an example of configuring access control permissions for Hive views:

- If the current_user function is not used, different views need to be created for different users to access different data.
  - Authorize the view **v1** permission to user **hiveuser1**. The user **hiveuser1** can access data with **type** set to **hiveuser1** in **table1**.

    create view v1 as select * from table1 where type='hiveuser1';
  - Authorize the view **v2** permission to user **hiveuser2**. The user **hiveuser2** can access data with **type** set to **hiveuser2** in **table1**.

    create view v2 as select * from table1 where type='hiveuser2';
- If the current_user function is used, only one view needs to be created.

  Authorize the view **v** permission to users **hiveuser1** and **hiveuser2**. When user **hiveuser1** queries view **v**, the current_user() function is automatically converted to **hiveuser1**. When user **hiveuser2** queries view **v**, the **current_user()** function is automatically converted to **hiveuser2**.

  create view v as select * from table1 where type=current_user();

----End

12.8.11 Allowing Users without ADMIN Permission to Create Temporary Functions

Scenario

You must have **ADMIN** permission when creating temporary functions on Hive of the open source community version.

MRS Hive provides a configuration switch. The default value is true, that is, creating temporary functions requires the **ADMIN** permission, which is consistent with the open source community version.

You can modify configurations of this function. After the function is enabled, you can create temporary functions without **ADMIN** permission. If this parameter is set to **false**, security risks exist.

NOTE

The security mode supports the configuration of whether the ADMIN permission is required for creating temporary functions, but the common mode does not support this function.

Procedure

Step 1 Log in to FusionInsight Manager. Click **Cluster**, choose **Services** > **Hive**, click **Configurations**, and then **All Configurations**.

Step 2 Enter the parameter name in the search box, search for **hive.security.temporary.function.need.admin**, change the parameter value to **true** or **false**, and restart all HiveServer instances.

NOTE

- If this parameter is set to **true**, the ADMIN permission is required for creating temporary functions, which is the same as that in the open source community.
- If this parameter is set to **false**, the ADMIN permission is not required for creating temporary functions.

----End

12.8.12 Allowing Users with Select Permission to View the Table Structure

Scenario

If the select permission is granted to a user during Hive table creation, the user can run the **show create table** command to view the table structure.

Procedure

Step 1 Log in to FusionInsight Manager. Click **Cluster**, choose **Services** > **Hive**, click **Configurations**, and then **All Configurations**.

Step 2 Choose **HiveServer(Role)** > **Customization**, add a customized parameter to the **hive-site.xml** parameter file, set **Name** to **hive.allow.show.create.table.in.select.nogrant**, and set **Value** to **true**.

Step 3 Click **Save** to save the configuration. Click **Instances**, select all Hive instances, choose **More** > **Restart Instance**, enter the password of the current user, and click **OK** to restart all Hive instances.

Step 4 Determine whether to enable this function on the Spark/Spark2x client.

- If yes, download and install the Spark/Spark2x client again.
- If no, no further action is required.

----End

12.8.13 Allowing Only the Hive Administrator to Create Databases and Tables in the Default Database

Scenario

Only the Hive administrator can create databases and tables in the default database. Other users can use the databases only after being authorized by the Hive administrator.

NOTE

- After this function is enabled, common users are not allowed to create a database or create a table in the default database. Based on the actual application scenario, determine whether to enable this function.
- Permissions of common users are restricted. In the scenario where common users have been used to perform operations, such as database creation, table script migration, and metadata recreation in an earlier version of database, the users can perform such operations on the database in the condition that this function is disabled temporarily after the database is migrated or after the cluster is upgraded.

Procedure

Step 1 Log in to FusionInsight Manager. Click **Cluster**, choose **Services** > **Hive**, click **Configurations**, and then **All Configurations**.

Step 2 Choose **HiveServer(Role)** > **Customization**, add a customized parameter to the **hive-site.xml** parameter file, set **Name** to **hive.allow.only.admin.create**, and set **Value** to **true**.

Step 3 Click **Save** to save the configuration. Click **Instances**, select all Hive instances, choose **More** > **Restart Instance**, enter the password of the current user, and click **OK** to restart all Hive instances.

Step 4 Determine whether to enable this function on the Spark/Spark2x client.

- If yes, go to **Step 5**.
- If no, no further action is required.

Step 5 Choose **SparkResource2x** > **Customization**, add a customized parameter to the **hive-site.xml** parameter file, set **Name** to **hive.allow.only.admin.create**, and set **Value** to **true**. Then, choose **JDBCServer2x** > **Customization** and repeat the preceding operations to add the customized parameter.

Step 6 Click **Save** to save the configuration. Click **Instances**, select all Spark2x instances, choose **More** > **Restart Instance**, enter the password of the current user, and click OK to restart all instances.

Step 7 Download and install the Spark/Spark2x client again.

----End

12.8.14 Configuring Hive to Support More Than 32 Roles

Scenario

The number of OS user groups is limited, and the number of roles that can be created in Hive cannot exceed 32. After this function is enabled, more than 32 roles can be created in Hive.

NOTE

- After this function is enabled and the table or database is authorized, roles that have the same permission on the table or database will be combined using vertical bars (|). When the ACL permission is queried, the combined result is displayed, which is different from that before the function is enabled. This operation is irreversible. Determine whether to make adjustment based on the actual application scenario.
- If the current component uses Ranger for permission control, you need to configure related policies based on Ranger for permission management. For details, see **Adding a Ranger Access Permission Policy for Hive**.
- After this function is enabled, a maximum of 512 roles (including **owner**) are supported by default. The number is controlled by the user-defined parameter **hive.supports.roles.max** of MetaStore. You can change the value based on the actual application scenario.

Procedure

Step 1 Log in to FusionInsight Manager. Click **Cluster**, choose **Services** > **Hive**, click **Configurations**, and then **All Configurations**.

Step 2 Modify parameters and restart related instances:

- For versions earlier than MRS 3.2.0:
  a. Choose **MetaStore(Role)** > **Customization**, add a custom parameter to the **hivemetastore-site.xml** parameter file, and set **Name** to **hive.supports.over.32.roles** and **Value** to **true**.
  b. Choose **HiveServer(Role)** > **Customization** for versions earlier than MRS 3.2.0, add a custom parameter to the **hive-site.xml** parameter file, set **Name** to **hive.supports.over.32.roles**, and set **Value** to **true**.
  c. Click **Save** to save the configuration.
  d. Click **Instances**, select all Hive instances, and choose **More** > **Restart Instance**, enter the password of the current user, and click **OK** to restart all Hive instances.
- For MRS 3.2.0 or later:
  a. Choose **MetaStore(Role)** > **Customization**, add a custom parameter to the **hivemetastore-site.xml** parameter file, and set **Name** to **hive.supports.over.32.roles** and **Value** to **true**.
  b. Click **Save** to save the configuration.
  c. Click **Instances**, select all MetaStore instances, and choose **More** > **Restart Instance**, enter the password of the current user, and click **OK** to restart all MetaStore instances.

----End

12.8.15 Creating User-Defined Hive Functions

When the built-in functions of Hive cannot meet requirements, you can compile user-defined functions (UDFs) and use them in queries.

According to implementation methods, UDFs are classified as follows:

- Common UDFs: used to perform operations on a single data row and export a single data row.

- User-defined aggregating functions (UDAFs): used to input multiple data rows and export a single data row.
- User-defined table-generating functions (UDTFs): used to perform operations on a single data row and export multiple data rows.

According to use methods, UDFs are classified as follows:

- Temporary functions: used only in the current session and must be recreated after a session restarts.
- Permanent functions: used in multiple sessions. You do not need to create them every time a session restarts.

NOTE

- You need to properly control the memory and thread usage of variables in UDFs. Improper control may cause memory overflow or high CPU usage.
- If Ranger authentication is enabled for the cluster, you need to disable Ranger authentication before using Python UDFs.

The following uses AddDoublesUDF as an example to describe how to compile and use UDFs.

Function

AddDoublesUDF is used to add two or more floating point numbers. In this example, you can learn how to write and use UDFs.

NOTE

- A common UDF must be inherited from **org.apache.hadoop.hive.ql.exec.UDF**.
- A common UDF must implement at least one **evaluate()**. The evaluate function supports overloading.
- To develop a UDF, add the **hive-exec-*.jar** dependency package to the project. You can obtain the package from the Hive service installation directory, for example, **${BIGDATA_HOME}/components/FusionInsight_HD_*/Hive/disaster/plugin/lib/**.

Sample Code

The following is a UDF code example:

xxx indicates the name of the organization that develops the program.

```
package com.xxx.bigdata.hive.example.udf;
import org.apache.hadoop.hive.ql.exec.UDF;

public class AddDoublesUDF extends UDF {
 public Double evaluate(Double... a) {
   Double total = 0.0;
   // Processing logic
   for (int i = 0; i < a.length; i++)
     if (a[i] != null)
       total += a[i];
   return total;
 }
}
```

Creating User-Defined Hive Functions

Step 1 Prepare the user who will execute the function.

1. Log in to Manager as user **admin**, choose **Cluster** > **Cluster Properties**, and check and record the authentication method of the cluster.
2. Choose **Cluster** > **Services** > **Hive**, click **More** in the upper right corner of the page, and check whether Ranger authentication is enabled for Hive.
3. Choose **System** > **Permission** > **User**, click **Create**, set the following parameters, and click **OK**.
   - **Username**: Enter a username, for example, **test**.
   - **User Type**: Select **Human-Machine**.
   - **Password** and **Confirm Password**: Enter a password and enter it again.
   - **User Group**: Click **Add**, select **hive** and **hadoop** groups, and click **OK**.
4. Assign permissions to the new user based on the cluster authentication method and whether Ranger authentication is enabled.
   - The cluster is in security mode.
     - If the **Enable Ranger** button is grayed out (Ranger authentication has been enabled for Hive), go to **Step 1.5**.
     - If the **Disable Ranger** button is grayed out (Ranger authentication has been disabled for Hive), go to **Step 1.6**.
   - The cluster is in normal mode.
     - If the **Enable Ranger** button is grayed out (Ranger authentication has been enabled for Hive), go to **Step 1.5**.
     - If the **Disable Ranger** button is grayed out (Ranger authentication has been disabled for Hive), go to **Step 2**.
5. Hive uses Ranger for authentication. Log in to the Ranger management page as the Ranger administrator (**rangeradmin** for security mode and **admin** for normal mode) to add Hive permission control policies for the user.
   a. Choose **Cluster** > **Services** > **Ranger** and click the hyperlink on the right of **Ranger web UI**.
   b. For a cluster in security mode, click the username in the upper right corner of the page, click **Log Out**. Log in to the Ranger management page as user **rangeradmin**.
   c. On the home page, click the component plug-in name in the **HADOOP SQL** area, for example, **Hive**.
   d. In the **Access** tab, click **Add New Policy**, set the following parameters, and click **Add**:
      - **Policy Name**: Set the policy name, for example, **test_hive**.
      - **database**
        - Permanent function: Enter the name of the database to which the function is to be added, for example, **default**.
        - Temporary function: Switch **database** to **global** and enter a specific function name or set it to *****.

- **table**: Switch to **udf** and enter a specific function name or set it to *. You do not need to set this parameter for temporary functions.
- In the **Allow Conditions** area, select the new user in the **Select User** column and add the following permissions to **Permissions**:
  - Permanent function: Grant permission based on service requirements. For example, you can add the **create**, **select**, and **drop** permissions.
  - Temporary function: Add the **Temporary UDF Admin** permission.

6. Hive uses the Manager role for authentication. Create a user with the Hive administrator permission to execute permanent and temporary functions.
   a. On the homepage of Manager, choose **System** > **Permission** > **Role**, click **Create Role**, set the following parameters, and click **OK**.
      - **Role Name**: Enter a role name, for example, **test_role**.
      - **Configure Resource Permission**: Click the name of the desired cluster, click **Hive**, and select **Admin**.
   b. Click **User** and click **Modify** in the row that contains the user created by **Step 1.3**.
   c. On the **Modify User** page, click **Add** on the right of **Role**, add the newly created role with the Hive administrator rights, and click **OK**.

Step 2 Package the preceding program into **AddDoublesUDF.jar** and upload it to the client installation node, for example, the **opt** directory. Then upload the package to a specified directory in HDFS, for example, **/user/hive_examples_jars**: Both the user who creates the function and the user who uses the function must have the read permission on the file.

1. Go to the client installation directory and configure the environment variables:

   cd *Client installation directory*

   source bigdata_env
2. Authenticate the user.
   - For a cluster with Kerberos authentication enabled (security mode):

     kinit *Service user*
   - For a cluster with Kerberos authentication disabled (normal mode):

     export HADOOP_USER_NAME= *Service user*
3. Upload the UDF JAR package to the HDFS directory.

   hdfs dfs -put */opt /user/hive_examples_jars*

   hdfs dfs -chmod 777 */user/hive_examples_jars*

Step 3 Log in to the Hive client.

- For a cluster with Kerberos authentication enabled (security mode), run the following command:

  beeline

NOTE

If the user is assigned the Hive administrator role, run the following command to switch to the **admin** role in each **beeline** maintenance operation session and then perform subsequent operations:

set role admin;

- For a cluster with Kerberos authentication enabled (security mode), run the following command:

  beeline -n *Hive service user*

Step 4 Run the following commands on the Hive Server to define the function:

- Create a permanent function.

  CREATE FUNCTION *addDoubles* **AS 'com.***xxx***.bigdata.hive.example.udf.AddDoublesUDF' using jar 'hdfs://hacluster/user/hive_examples_jars/AddDoublesUDF.jar';**

  addDoubles is the alias of the function, which is used in **SELECT** queries. *xxx* is typically the name of the organization that develops the program.

- Create a temporary function.

  CREATE TEMPORARY FUNCTION addDoubles AS 'com.*xxx***.bigdata.hive.example.udf.AddDoublesUDF' using jar 'hdfs://hacluster/user/hive_examples_jars/AddDoublesUDF.jar';**

  - *addDoubles* indicates the function alias that is used for SELECT query.
  - **TEMPORARY** indicates that the function is used only in the current session of the Hive Server.

Step 5 Run the following command on the Hive Server to use the function:

SELECT addDoubles(1,2,3);

NOTE

If an [Error 10011] error is displayed when you log in to the client again, run the **reload function;** command and then use this function.

Step 6 Run the following command on the Hive Server to delete the function:

DROP FUNCTION addDoubles;

----End

Extended Applications

None

12.8.16 Configuring High Reliability for Hive Beeline

Scenario

- When the beeline client is disconnected due to network exceptions during the execution of a batch processing task, tasks submitted before beeline is disconnected can be properly executed in Hive. When you start the batch processing task again, the submitted tasks are not executed and tasks that are not executed are executed in sequence.

- When the HiveServer service breaks down due to some reasons during the execution of a batch processing task, Hive enables that the tasks that have been successfully executed are not executed again when the same batch processing task is started again. The execution starts from the task that has not been executed from the time when HiveServer2 breaks down.

Example

1. Log in to the node where the Hive client is installed as the Hive client installation user.
2. Run the following commands to switch to the client installation directory, configure environment variables, and authenticate users:

   cd *Client installation directory*

   source bigdata_env

   kinit *Hive service user* (Skip this step if Kerberos authentication is not enabled for the cluster.)
3. Beeline is reconnected after being disconnection.

   Example:

   beeline -e "${*SQL*}" **--hivevar batchid=***xxx*
4. Beeline kills the running tasks.

   Example:

   beeline -e "" --hivevar batchid=*xxxxx* **--hivevar kill=true**
5. Log in to the beeline client and start the mechanism of reconnection after disconnection.

   beeline

   set hivevar:batchid=*xxx*

NOTE

- *xxxx* indicates the batch ID of tasks submitted in the same batch using the beeline client. Batch IDs can be used to identify the task submission batch. If the batch ID is not contained when a task is submitted, this feature is not enabled. The value of *xxxx* is specified during task execution. In the following example, the value of *xxxx* is **012345678901**.

  beeline -f hdfs://hacluster/user/hive/table.sql --hivevar batchid=012345678901

- If the running SQL script depends on the data timeliness, you are advised not to enable the breakpoint reconnection mechanism. You can use a new batch ID to submit tasks. During reexecution of the scripts, some SQL statements have been executed and are not executed again. As a result, expired data is obtained.
- If some built-in time functions are used in the SQL script, it is recommended that you do not enable the breakpoint reconnection mechanism or the use of a new batch ID for each execution. The reason is the same as above.
- A SQL script contains one or more subtasks. If the logic for deleting and creating temporary tables exist in the SQL script, it is recommended that the logic for deleting temporary tables be placed at the end of the script. If the subtasks executed after the temporary table deletion task fail to be executed and the temporary table is used in the subtasks before the temporary table deletion task, when the SQL script is executed using the same batch ID for the next time, the compilation of the subtasks (excluding the task for creating the temporary table because the creation has been completed and is not executed again, and only compilation is allowed) executed before the temporary table deletion task fails because the temporary has been deleted. In this case, you are advised to use a new batch ID to execute the script.

Parameter description:

- **zk.cleanup.finished.job.interval**: indicates the interval for executing the cleanup task. The default interval is 60 seconds.
- **zk.cleanup.finished.job.outdated.threshold**: indicates the threshold of the node validity period. A node is generated for tasks in the same batch. The threshold is calculated from the end time of the execution of the current batch task. If the time exceeds 60 minutes, the node is deleted.
- **batch.job.max.retry.count**: indicates the maximum number of retry times of a batch task. If the number of retry times of a batch task exceeds the value of this parameter, the task execution record is deleted. The task will be executed from the first task when the task is started next time. The default value is **10**.
- **beeline.reconnect.zk.path**: indicates the root node for storing task execution progress. The default value for the Hive service is **/beeline**.

12.8.17 Detecting Statements That Overwrite a Table with Its Own Data

Scenario

You can intercept SQL statements that read the data written by themselves in Hive. You can add the rule **dynamic_0004** on the **SQL Inspector** page to intercept SQL statements that overwrite a table, partition, or directory with data from the same table, partition, or directory.

This topic is available for MRS 3.5.0 or later versions.

NOTE

The following are some typical large Hive SQL statements you need to detect:

- Non-partitioned table

  INSERT OVERWRITE TABLE *tbl_a* **SELECT * FROM** *tbl_a*;

- Static partition table

  INSERT OVERWRITE TABLE *tbl_b* **PARTITION (ds='20240101') SELECT * FROM** *tbl_b* **WHERE ds='20240101';**

- Dynamic partitioning

  INSERT OVERWRITE TABLE *tbl_c* **PARTITION(pday) SELECT id,name,pday FROM** *tbl_c* **WHERE id > 100;**

- Writing data to an HDFS table

  INSERT OVERWRITE DIRECTORY 'hdfs://*hacluster/user/hive/warehouse/tbl_d***' SELECT * FROM** *tbl_d*;

Procedure

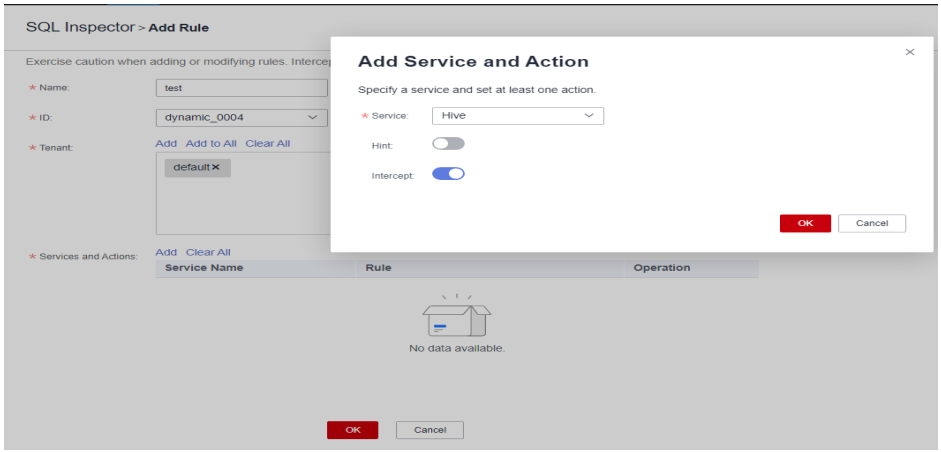
Step 1 Log in to FusionInsight Manager as a user with management right.

Step 2 Choose **Cluster** > **SQL Inspector**, click **Add Rule**, enter the password of the user, and click **OK**. The **Add Rule** page is displayed.

Step 3 On the displayed page, set the following parameters and click **OK**:

- **Name**: Enter a rule name, for example, **test**.
- ID: Select **dynamic_0004**.
- **Tenant**: Click **Add** and select the tenant name for which the inspection rule will be used, for example, **default**.
- **Services and Actions**: Click **Add**. In the **Add Service and Action** dialog box, set the following parameters and click **OK**:
  - **Service**: Select **Hive**.
  - Enable the **Hint** or **Intercept** action based on service requirements.

Figure 12-4 Creating a Hive SQL inspection rule



Step 4 Log in to the node where the Hive client is installed and switch to the client installation directory.

cd /opt/client

Configure environment variables.

source bigdata_env

Authenticate the user (skip this step if Kerberos authentication is disabled for the cluster).

kinit *Component service user who has the Hive operation permission*

Step 5 Log in to the Hive client.

beeline

Step 6 Create a table and insert data into the table.

drop table *tbl_a***;**

create table *tbl_a***(id int, name string);**

insert into table *tbl_a* **values(123,'sjk'),(234,'shen'),(111,'aaa');**

Step 7 (Optional) Enable **strict** interception. For details, see **Setting Strict Interception for Hive Dynamic Partitioned Tables**. The **nonstrict** mode is used by default for dynamic partitioning.

set hive-ext.dynamic.partition.intercept.mode=strict;

Step 8 Check whether the **static_0001** rule has been applied.

insert overwrite table *tbl_a* **select * from** *tbl_a***;**

If the configured action is **Hint**, the system reports an alarm when detecting a SQL statement that triggers the inspection rule. The SQL statement will continue to run, and the following information will be displayed:

```
WARN  : DYNAMIC_0004 Self-read and self-overwrite operations are not allowed.
```

If the configured action is **Intercept**, the SQL statement will be intercepted when the system detects a SQL statement that triggers the inspection rule. The following information is displayed:

```
Error: Error while compiling statement: FAILED: RuleException DYNAMIC_0004 Self-read and self-overwrite operations are not allowed. (state=42000,code=40000)
```

----End

Setting Strict Interception for Hive Dynamic Partitioned Tables

The system cannot obtain the partition information in the compilation phase of dynamic partitioning, and therefore cannot detect SQL statements that overwrite data with the same data. You can choose a non-strict or strict interception policy when SQL statements involve dynamic partitioning.

- Non-strict interception allows the system to check whether the queried table is the one where the submitted SQL statement writes data to. The detection is completed during SQL compilation. So, non-strict interception is fast, but it also works for quires of data in different partitions.
- Strict interception allows the system to check whether the queried partition is the one where the submitted SQL statement writes data to. The detection is completed during the MoveTask phase of a SQL task. Strict interception is

more accurate, but it works only after most logics of the SQL task are executed, causing resource waste.

Non-strict interception is used by default for dynamic partitioning. To enable strict interception, set **hive-ext.lakeformation.transform.role** (in **Table 12-9**) to **strict**. You are advised to set this parameter for a single task, for example, for a SQL statement like **set hive-ext.dynamic.partition.intercept.mode=strict;**.

Table 12-9 Parameters

Parameter	Default Value	Description
hive-ext.dynamic.partition.intercept.mode	nonstrict	Interception mode for SQL statements that involve Hive dynamic partitioning. The options are as follows: • **nonstrict**: If the queried table is the one where a SQL statement inserts data to, the SQL statement is intercepted. • **strict**: If the queried partition is the one where a SQL statement inserts data to, the SQL statement is intercepted.

12.8.18 Configuring Hive Dynamic Data Masking

Scenarios

Enabling Hive dynamic masking allows for the utilization of data within the masked column for computations, while keeping it concealed during the output of calculation results. The cluster's masking policy is dynamically transferred in accordance with lineage relationships, optimizing data utility while safeguarding privacy.

Constraints

- Data masking is not available for Hudi tables.
- Masking for direct HDFS read/write operations is not supported.
- Masking for complex data types like arrays, maps, and structs is not supported.
- Custom masking policies support only the string type, and the values are masked by ***.
- In instances where the masking policy transfer results in a conflict with an existing policy on the target table, the latter's policy will be overridden as **Custom:"***"**.
- For simple queries that are not submitted in a YARN job, the masking result complies with the masking policy configured on Ranger. With the customer-

type masking policy, data is masked by ***. Simple queries include **select * from** *Table name*; and **select * from** *Table name* **limit** *xxx*;.

- For complex queries that are submitted in a Yarn job, string fields are masked in compliance with the masking policy configured on Ranger. Other types are masked based on the Nullify masking policy.

Configuring Hive Dynamic Data Masking

Step 1 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Hive**, and click **Configurations**. Search for **hive.dynamic.masked.enabled** and change the value of this parameter for the HiveServer instance to **true**.

Step 2 Click **Save**. Click the **Instances** tab, select all HiveServer instances, click **More** > **Restart Instance**, enter the user password, and click **OK** to restart all HiveServer instances.

Step 3 Log in to the node where the Hive client is installed as the client installation user and run the following commands:

cd *Client installation directory*

source bigdata_env

source Hive/component_env

kinit *Component service user* (skip this step if Kerberos authentication is disabled for the cluster (the cluster is in normal mode))

Step 4 Log in to the Hive client and create a Hive table.

beeline

create table *hivetest(a int, b string)*;

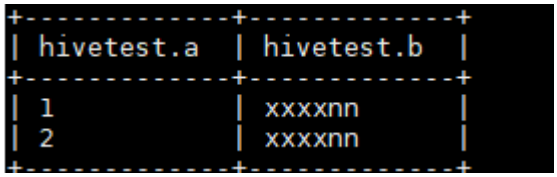
insert into hivetest **values** *(1,"test01"), (2,"test02")*;

Step 5 Configure a masking policy for field **b** in the **hivetest** table by referring to **Hive Data Masking** and check the masking result.

select * from *hivetest*;

If the following information is displayed, data masking is successful.

Figure 12-5 Successful data masking



Step 6 Verify the transferability of the masking policy.

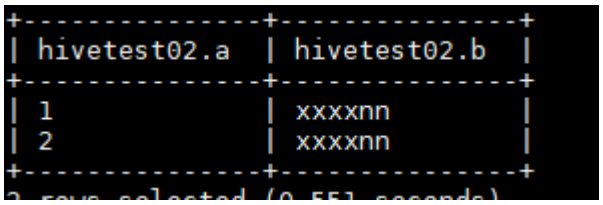
create table *hivetest02* **as select * from** *hivetest*;

Wait for approximately 1 minute, the Ranger policy is synchronized to the Hive component. Check whether the masking policy is transferred.

select * from *hivetest02*;

If the following information is displayed, the masking policy is successfully transferred.

Figure 12-6 Masking policy transferred



Step 7 If the new table data in **Step 6** is successfully masked, the dynamic masking configuration has been applied. Log in to the Ranger management page as user **rangeradmin**, click **Hive** in the **HADOOP SQL** area on the home page, and click the **Masking** tab. Check the automatically generated masking policy of the table. For example, the masking policy of table **hivetest02** is **policy_synced_from_table_default_hivetest02_b**.

Figure 12-7 Masking policy of the table

----End

12.9 Hive Performance Tuning

12.9.1 Creating Hive Table Partitions to for Faster Queries

Scenario

During the Select query, Hive generally scans the entire table, which is time-consuming. To improve query efficiency, create table partitions based on service requirements and query dimensions.

Procedure

Step 1 Log in to the node where the Hive client has been installed as user **root**.

Step 2 Run the following command to go to the client installation directory, for example, **/opt/client**:

cd /opt/client

Step 3 Run the **source bigdata_env** command to configure environment variables for the client.

Step 4 Run the following command on the client for login:

kinit *Username*

Step 5 Run the following command to log in to the Hive client:

beeline

Step 6 Select the static or dynamic partition.

- Static partition:

  Manually enter a partition name, and use the keyword **PARTITIONED BY** to specify partition column name and data type when creating a table. During application development, use the **ALTER TABLE ADD PARTITION** statement to add a partition and use the **LOAD DATA INTO PARTITION** statement to load data to the partition, which supports only static partitions.

- Dynamic partition: Use a query command to insert results to a partition of a table. The partition can be a dynamic partition.

  The dynamic partition can be enabled on the client tool by running the following command:

  set hive.exec.dynamic.partition=true;

  The default mode of the dynamic partition is **strict**. That is, at least a column must be specified as a static partition, under which dynamic sub-partitions can be created. You can run the following command to enable a completely dynamic partition:

  set hive.exec.dynamic.partition.mode=nonstrict;

NOTE

- The dynamic partition may cause a DML statement to create a large number of partitions and new mapping folders, which deteriorates system performance.
- If there are a large number of files, it takes a long time to run a SQL statement. You can run the **set mapreduce.input.fileinputformat.list-status.num-threads = 100;** statement before running a SQL statement to shorten the time. The parameter **mapreduce.input.fileinputformat.list-status.num-threads** can be set only after being added to the Hive whitelist.

----End

12.9.2 Optimizing Hive Joins

Scenario

When the Join statement is used, the command execution speed and query speed may be slow in case of large data volume. To resolve this problem, you can optimize Join.

Join optimization can be classified into the following modes:

- Map Join
- Sort Merge Bucket Map Join
- Optimizing Join Sequences

Map Join

Hive Map Join applies to small tables (the table size is less than 25 MB) that can be stored in the memory. The table size can be defined using **hive.mapjoin.smalltable.filesize**, and the default table size is 25 MB.

Map Join has two methods:

- Use /*+ MAPJOIN(join_table) */.

- Set the following parameter before running the statement. The default value is **true** in the current version.

  set hive.auto.convert.join=true;

There is no Reduce task when Map Join is used. Instead, a MapReduce Local Task is created before the Map job. The task uses TableScan to read small table data to the local computer, saves and writes the data in HashTable mode to a hard disk on the local computer, upload the data to DFS, and saves the data in distributed cache. The small table data that the map task reads from the local disk or distributed cache is the output together with the large table join result.

When using Map Join, make sure that the size of small tables cannot be too large. If small tables use up memory, the system performance will deteriorate and even memory leakage occurs.

Sort Merge Bucket Map Join

The following conditions must be met before using Sort Merge Bucket Map Join:

- The two Join tables are large and cannot be stored in the memory.
- The two tables are bucketed (clustered by (column)) and sorted (sorted by(column)) according to the join key, and the buckets counts of the two tables are in integral multiple relationship.

Set the following parameters to enable Sort Merge Bucket Map Join:

set hive.optimize.bucketmapjoin=true;

set hive.optimize.bucketmapjoin.sortedmerge=true;

This type of Map Join does not have Reduce tasks too. A MapReduce Local Task is started before the Map job to read small table data by bucket to the local computer. The local computer saves the HashTable backup of multiple buckets and writes the backup into HDFS. The backup is also saved in the distributed cache. The small table data that the map task reads from the local disk or distributed cache by bucket is the output after mapping with the large table.

Optimizing Join Sequences

If the Join operation is to be performed on three or more tables and different Join sequences are used, the execution time will be greatly different. Using an appropriate Join sequence can shorten the time for task execution.

Rules of a Join sequence:

- A table with small data volume or a combination with fewer results generated after a Join operation is executed first.
- A table with large data volume or a combination with more results generated after a Join operation is executed later.

For example, the **customer** table has the largest data volume, and fewer results will be generated if a Join operation is performed on the **orders** and **lineitem** tables first.

The original Join statement is as follows.

```
select
  l_orderkey,
```



```
  sum(l_extendedprice * (1 - l_discount)) as revenue,
  o_orderdate,
  o_shippriority
from
  customer,
  orders,
  lineitem
where
  c_mktsegment = 'BUILDING'
  and c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and o_orderdate < '1995-03-22'
  and l_shipdate > '1995-03-22'
limit 10;
```

After the sequence is optimized, the Join statements are as follows:

```
select
  l_orderkey,
  sum(l_extendedprice * (1 - l_discount)) as revenue,
  o_orderdate,
  o_shippriority
from
  orders,
  lineitem,
  customer
where
  c_mktsegment = 'BUILDING'
  and c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and o_orderdate < '1995-03-22'
  and l_shipdate > '1995-03-22'
limit 10;
```

Precautions

Join Data Skew Problem

Data skew refers to the symptom that the task progress is 99% for a long time.

Data skew often exists because the data volume of a few Reduce tasks is much larger than that of others. Most Reduce tasks are complete while a few Reduce tasks are not complete.

To resolve the data skew problem, set **hive.optimize.skewjoin=true** and adjust the value of **hive.skewjoin.key**. **hive.skewjoin.key** specifies the maximum number of keys received by a Reduce task. If the number reaches the maximum, the keys are atomically distributed to other Reduce tasks.

12.9.3 Optimizing the Hive Group By Statement

Scenario

Optimize the Group by statement to accelerate the command execution and query speed.

During the Group by operation, Map performs grouping and distributes the groups to Reduce; Reduce then performs grouping again. Group by optimization can be performed by enabling Map aggregation to reduce Map output data volume.

Procedure

On a Hive client, set the following parameter:


```
set hive.map.aggr=true
```

Precautions

- **Group By Data Skew**

  Group by have data skew problems. When **hive.groupby.skewindata** is set to **true**, the created query plan has two MapReduce jobs. The Map output result of the first job is randomly distributed to Reduce tasks, and each Reduce task performs aggregation operations and generates output result. Such processing may distribute the same Group By Key to different Reduce tasks for load balancing purpose. According to the preprocessing result, the second Job distributes Group By Key to Reduce to complete the final aggregation operation.

- **Count Distinct Aggregation Problem**

  When the aggregation function count distinct is used in deduplication counting, serious Reduce data skew occurs if the processed value is empty. The empty value can be processed independently. If count distinct is used, exclude the empty value using the where statement and increase the last count distinct result by 1. If there are other computing operations, process the empty value independently and then combine the value with other computing results.

12.9.4 Optimizing Hive OCR Data Storage

Scenario

ORC is an efficient column storage format and has higher compression ratio and reading efficiency than other file formats.

You are advised to use **ORC** as the default Hive table storage format.

Prerequisites

You have logged in to the Hive client. For details, see **Using the Hive Client**.

Procedure

- Recommended: **SNAPPY** compression, which applies to scenarios with even compression ratio and reading efficiency requirements.

  Create table *xx (col_name data_type)* stored as orc tblproperties ("orc.compress"="SNAPPY");

- Available: **ZLIB** compression, which applies to scenarios with high compression ratio requirements.

  Create table *xx (col_name data_type)* stored as orc tblproperties ("orc.compress"="ZLIB");

NOTE

xx indicates the specific Hive table name.

12.9.5 Optimizing Hive SQL Logic

Scenario

When SQL statements are executed on Hive, if the **(a&b) or (a&c)** logic exists in the statements, you are advised to change the logic to **a & (b or c)**.

Example

If condition a is **p_partkey = l_partkey**, the statements before optimization are as follows:

```
select
	sum(l_extendedprice* (1 - l_discount)) as revenue
from
	lineitem,
	part
where
	(
		p_partkey = l_partkey
		and p_brand = 'Brand#32'
		and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
		and l_quantity >= 7 and l_quantity <= 7 + 10
		and p_size between 1 and 5
		and l_shipmode in ('AIR', 'AIR REG')
		and l_shipinstruct = 'DELIVER IN PERSON'
	)
	or
	(	p_partkey = l_partkey
		and p_brand = 'Brand#35'
		and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
		and l_quantity >= 15 and l_quantity <= 15 + 10
		and p_size between 1 and 10
		and l_shipmode in ('AIR', 'AIR REG')
		and l_shipinstruct = 'DELIVER IN PERSON'
	)
	or
	(	p_partkey = l_partkey
		and p_brand = 'Brand#24'
		and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
		and l_quantity >= 26 and l_quantity <= 26 + 10
		and p_size between 1 and 15
		and l_shipmode in ('AIR', 'AIR REG')
		and l_shipinstruct = 'DELIVER IN PERSON'
	)
```

The statements after optimization are as follows:

```
select
	sum(l_extendedprice* (1 - l_discount)) as revenue
from
	lineitem,
	part
where   p_partkey = l_partkey and
	((
		p_brand = 'Brand#32'
		and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
		and l_quantity >= 7 and l_quantity <= 7 + 10
		and p_size between 1 and 5
		and l_shipmode in ('AIR', 'AIR REG')
		and l_shipinstruct = 'DELIVER IN PERSON'
	)
	or
	(
		p_brand = 'Brand#35'
		and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
```



```
        and l_quantity >= 15 and l_quantity <= 15 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
)
or
(
        p_brand = 'Brand#24'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 26 and l_quantity <= 26 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
))
```

12.9.6 Optimizing the Multi-Table Queries with Hive CBO

Scenario

When joining multiple tables in Hive, Hive supports Cost-Based Optimization (CBO). The system automatically selects the optimal plan based on the table statistics, such as the data volume and number of files, to improve the efficiency of joining multiple tables. Hive needs to collect table statistics before CBO optimization.

NOTE

- The CBO optimizes the joining sequence based on statistics and search criteria. However, the joining sequence may fail to be optimized in some special scenarios, such as data skew occurs and query condition values are not in the table.
- When column statistics collection is enabled, Reduce operations must be performed for aggregation. For insert tasks without the Reduce phase, Reduce operations will be performed to collect statistics.

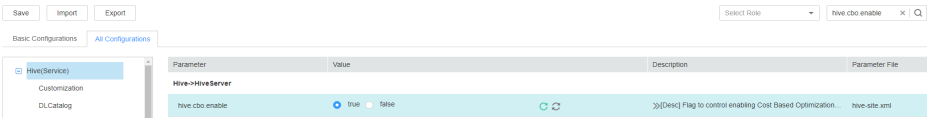
Prerequisites

You have logged in to the Hive client. For details, see **Using the Hive Client**.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Hive** > **Configurations**.

Step 2 Search for **hive.cbo.enable** in the search box, change the value to true to permanently enable the CBO function, save the configuration, and restart the affected instance for the configuration to take effect.



Step 3 Collect statistics about the existing data in Hive tables manually.

Run the following command to manually collect statistics: Statistics about only one table can be collected. If statistics about multiple tables need to be collected, the command needs to be executed repeatedly.

ANALYZE TABLE [db_name.]tablename [PARTITION(partcol1[=val1], partcol2[=val2], ...)]

COMPUTE STATISTICS

[FOR COLUMNS]

[NOSCAN];

NOTE

- When **FOR COLUMNS** is specified, column-level statistics are collected.
- When NOSCAN is specified, statistics about the file size and number of files will be collected, but specific files will not be scanned.

For example:

analyze table table_name compute statistics;

analyze table table_name compute statistics for columns;

Step 4 Configure the automatic statistics collection function of Hive. After the function is enabled, new statistics will be collected only when you insert data by running the **insert overwrite/into** command.

- Run the following commands on the Hive client to enable the statistics collection function temporarily:

  To enable the automatic collection of table/partition-level statistics:

  set hive.stats.autogather = true;

  To enable the automatic collection of column-level statistics:

  set hive.stats.column.autogather = true;

  NOTE

  - The column-level statistics collection does not support complex data types, such as Map and Struct.
  - The automatic table-level statistics collection does not support Hive on HBase tables.

- On the Manager UI, search for the **hive.stats.autogather** and **hive.stats.column.autogather** parameters in the service configuration of Hive, and change the values to **true** to enable the collection function permanently.

Step 5 Run the following command to view statistics:

DESCRIBE FORMATTED table_name[.column_name] PARTITION partition_spec;

For example:

desc formatted table_name;

desc formatted table_name id;

desc formatted table_name partition(time='2016-05-27');

NOTE

Partition tables only support partition-level statistics collection, so you must specify partitions to query statistics for partition tables.

----End

12.10 Hive O&M Management

12.10.1 Hive Common Configuration Parameters

Hive is a data warehouse framework built on Hadoop. It provides the batch processing computing capability of the big data platform and can analyze and summarize structured and semi-structured data in batches to complete data computing.

This section describes common Hive parameters.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Hive** > **Configurations** > **All Configurations**.

Step 2 Search for the parameter name in the search box in the upper right corner and modify the parameter value. **Table 12-10**describes common Hive parameters.

Table 12-10 Hive parameter description

Parameter	Description	Default Value
hive.auto.convert.join	Whether Hive converts common **join** to **mapjoin** based on the input file size. **NOTE** When Hive is used to query a join table, whatever the table size is (if the data in the join table is less than 24 MB, it is a small one), set this parameter to **false**. If this parameter is set to **true**, new **mapjoin** cannot be generated when you query a join table.	Possible values are as follows: • true • false The default value is **true**.
hive.default.fileformat	Indicates the default file format used by Hive.	RCFile
hive.exec.reducers.max	Indicates the maximum number of reducers in a MapReduce job submitted by Hive.	999
hive.server2.thrift.max.worker.threads	Indicates the maximum number of threads that can be started in the HiveServer internal thread pool.	1,000

Parameter	Description	Default Value
hive.server2.thrift.min.worker.threads	Indicates the number of threads started during initialization in the HiveServer internal thread pool.	5
hive.hbase.delete.mode.enabled	Indicates whether to enable the function of deleting HBase records from Hive. If this function is enabled, you can use **remove table xx where xxx** to delete HBase records from Hive.	true
hive.metastore.server.min.threads	Indicates the number of threads started by MetaStore for processing connections. If the number of threads is more than the set value, MetaStore always maintains a number of threads that is not lower than the set value, that is, the number of resident threads in the MetaStore thread pool is always higher than the set value.	200
hive.server2.enable.doAs	Indicates whether to simulate client users during sessions between HiveServer2 and other services (such as Yarn and HDFS). If you change the configuration item from **false** to **true**, users with only the column permission lose the permissions to access corresponding tables.	true

Step 3 Click **Save** to save the configuration.

Step 4 Click **Instances**, select the corresponding instance, and choose **More** > **Restart Instance** for the configuration to take effect.

----End

12.10.2 Hive Log Overview

Log Description

Log path: The default save path of Hive logs is **/var/log/Bigdata/hive/***role name*, the default save path of Hive1 logs is **/var/log/Bigdata/hive1/***role name*, and the others follow the same rule.

- HiveServer: **/var/log/Bigdata/hive/hiveserver** (run log) and **var/log/Bigdata/audit/hive/hiveserver** (audit log)
- MetaStore: **/var/log/Bigdata/hive/metastore** (run log) and **/var/log/Bigdata/audit/hive/metastore** (audit log)

- WebHCat: **/var/log/Bigdata/hive/webhcat** (run log) and **/var/log/Bigdata/audit/hive/webhcat** (audit log)

Log archive rule: The automatic compression and archiving function of Hive is enabled. By default, when the size of a log file exceeds 20 MB (which is adjustable), the log file is automatically compressed. The naming rule of a compressed log file is as follows: <*Original log name*>-<*yyyy-mm-dd_hh-mm-ss*>.[*ID*].**log.zip** A maximum of 20 latest compressed files are reserved. The number of compressed files and compression threshold can be configured.

Table 12-11 Hive log list

Log Type	Log File Name	Description
Run log	/hiveserver/hiveserver.out	Log file that records HiveServer running environment information.
	/hiveserver/hive.log	Run log file of the HiveServer process.
	/hiveserver/hive-omm-*<Date>*-*<PID>*-gc.log.*<No.>*	GC log file of the HiveServer process.
	/hiveserver/prestartDetail.log	Work log file before the HiveServer startup.
	/hiveserver/check-serviceDetail.log	Log file that records whether the Hive service starts successfully
	/hiveserver/cleanupDetail.log	Cleanup log file about the HiveServer uninstallation
	/hiveserver/startDetail.log	Startup log file of the HiveServer process.
	/hiveserver/stopDetail.log	Shutdown log file of the HiveServer process.
	/hiveserver/localtasklog/omm_*<Date>*_*<Task ID>*.log	Run log file of the local Hive task.
	/hiveserver/localtasklog/omm_*<Date>*_*<Task ID>*-gc.log.*<No.>*	GC log file of the local Hive task.
	/metastore/metastore.log	Run log file of the MetaStore process.
	/metastore/hive-omm-*<Date>*-*<PID>*-gc.log.*<No.>*	GC log file of the MetaStore process.

Log Type	Log File Name	Description
	/metastore/postinstallDetail.log	Work log file after the MetaStore installation.
	/metastore/prestartDetail.log	Work log file before the MetaStore startup
	/metastore/cleanupDetail.log	Cleanup log file of the MetaStore uninstallation
	/metastore/startDetail.log	Startup log file of the MetaStore process.
	/metastore/stopDetail.log	Shutdown log file of the MetaStore process.
	/metastore/metastore.out	Log file that records MetaStore running environment information.
	/webhcat/webhcat-console.out	Log file that records the normal start and stop of the WebHCat process.
	/webhcat/webhcat-console-error.out	Log file that records the start and stop exceptions of the WebHCat process.
	/webhcat/prestartDetail.log	Work log file before the WebHCat startup.
	/webhcat/cleanupDetail.log	Cleanup logs generated during WebHCat uninstallation or before WebHCat installation
	/webhcat/hive-omm-<*Date*>-<PID>-gc.log.<*No.*>	GC log file of the WebHCat process.
	/webhcat/webhcat.log	Run log file of the WebHCat process
Audit log	hive-audit.log hive-rangeraudit.log	HiveServer audit log file
	metastore-audit.log	MetaStore audit log file.
	webhcat-audit.log	WebHCat audit log file.
	jetty-<Date>.request.log	Request logs of the jetty service.

Log Levels

Table 12-12 describes the log levels supported by Hive.

Levels of run logs are ERROR, WARN, INFO, and DEBUG from the highest to the lowest priority. Run logs of equal or higher levels are recorded. The higher the specified log level, the fewer the logs recorded.

Table 12-12 Log levels

Level	Description
ERROR	Logs of this level record error information about system running.
WARN	Logs of this level record exception information about the current event processing.
INFO	Logs of this level record normal running status information about the system and events.
DEBUG	Logs of this level record the system information and system debugging information.

To modify log levels, perform the following operations:

Step 1 Go to the **All Configurations** page of the Yarn service by referring to **Modifying Cluster Service Configuration Parameters**.

Step 2 On the menu bar on the left, select the log menu of the target role.

Step 3 Select a desired log level and save the configuration.

NOTE

The Hive log level takes effect immediately after being configured. You do not need to restart the service.

----End

Log Formats

The following table lists the Hive log formats:

Table 12-13 Log formats

Log Type	Format	Example
Run log	<yyyy-MM-dd HH:mm:ss,SSS>| <LogLevel>|<Thread that generates the log>| <Message in the log>| <Location of the log event>	2014-11-05 09:45:01,242 | INFO | main | Starting hive metastore on port 21088 | org.apache.hadoop.hive.metastore.HiveMetaStore.main(HiveMetaStore.java:5198)

Log Type	Format	Example
Audit log	<yyyy-MM-dd HH:mm:ss,SSS>|<LogLevel>|<Thread that generates the log>|<User Name><User IP><Time><Operation><Resource><Result><Detail >|<Location of the log event >	2018-12-24 12:16:25,319 | INFO | HiveServer2-Handler-Pool: Thread-185 | UserName=hive UserIP=10.153.2.204 Time=2018/12/24 12:16:25 Operation=CloseSession Result=SUCCESS Detail= | org.apache.hive.service.cli.thrift.ThriftCLIService.logAuditEvent(ThriftCLIService.java:434)

12.10.3 Importing and Exporting Hive Databases

Scenario

In big data application scenarios, Hive databases and all tables in these databases are usually migrated to another cluster. You can run the Hive database **export** and **import** commands to migrate a complete database.

NOTE

This section applies to MRS 3.2.0 or later.

The Hive database import and export function does not support importing or exporting encrypted tables, HBase external tables, Hudi tables, view tables, and materialized view tables.

Prerequisites

- If Hive databases are imported or exported across clusters and Kerberos authentication is enabled for both the source and destination clusters, configure cross-cluster mutual trust.
- If you want to run the **dump** or **load** command to import or export databases created by other users, grant the corresponding database permission to the users.
  - If Ranger authentication is not enabled for the cluster, log in to FusionInsight Manager to grant the administrator rights of the role to which the user belongs. For details, see section **Creating a Hive Role**.
  - If Ranger authentication is enabled for the cluster, grant users the permission to dump and load databases. For details, see **Adding a Ranger Access Permission Policy for Hive**.
- Enable the inter-cluster copy function in the source cluster and destination cluster.
- Configure the HDFS service address parameter for the source cluster to access the destination cluster.

  Log in to FusionInsight Manager of the source cluster, click **Cluster**, choose **Services** > **Hive**, and click **Configuration**. On the displayed page, search for **hdfs.site.customized.configs**, add custom parameter **dfs.namenode.rpc-**

address.haclusterX, and set its value to *Service IP address of the active NameNode instance node in the destination cluster:RPC port*. Add custom parameter **dfs.namenode.rpc-address.haclusterX1** and set its value to *Service IP address of the standby NameNode instance node in the destination cluster:RPC port*. The RPC port of NameNode is **25000** by default. After saving the configuration, roll-restart the Hive service.

Procedure

Step 1 Log in to the node where the client is installed in the source cluster as the Hive client installation user.

Step 2 Run the following command to switch to the client installation directory, for example, **/opt/client**:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 If Kerberos authentication is enabled for the cluster, run the following command to authenticate the user. Otherwise, skip this step.

kinit *Hive service user*

Step 5 Run the following command to log in to the Hive client:

beeline

Step 6 Run the following command to create the **dump_db** database:

create database *dump_db***;**

Step 7 Run the following command to switch to the **dump_db** database:

use *dump_db***;**

Step 8 Run the following command to create the **test** table in the **dump_db** database:

create table *test***(***id int***);**

Step 9 Run the following command to insert data to the **test** table:

insert into *test* **values(***123***);**

Step 10 Run the following command to set the **dump_db** database as the source of the replication policy:

alter database *dump_db* **set dbproperties (**'*repl.source.for*'='replpolicy1'**);**

NOTE

- Perform the following steps to set permissions for users when the **alter** command is used to modify database attributes:
  - If Ranger authentication is not enabled for the cluster, log in to FusionInsight Manager to grant the administrator rights of the role to which the user belongs. For details, see section **Creating a Hive Role**.
  - If Ranger authentication is enabled for the cluster, grant users the permission to dump and load databases. For details, see **Adding a Ranger Access Permission Policy for Hive**.
- Databases with replication policy sources configured can be deleted only after their replication policy sources are set to null. To do so, run the following command:

  alter database *dump_db* **set dbproperties ('***repl.source.for***'='');**

Step 11 Run the following command to export the **dump_db** database to the **/user/hive/test** directory of the destination cluster:

repl dump *dump_db* **with ('hive.repl.rootdir'='hdfs://hacluster***X***/user/hive/test');**

NOTE

- **hacluster *X*** is the value of **haclusterX** in new custom parameter**dfs.namenode.rpc-address.haclusterX**.
- Ensure that the current user has the read and write permissions on the export directory to be specified.

Step 12 Log in to the node where the client is installed in the destination cluster as the Hive client installation user, and perform **Step 2** to **Step 5**.

Step 13 Run the following command to import data from the **dump_db** database in the **/user/hive/test** directory to the **load_db** database:

repl load *load_db* **from** '*/user/hive/repl*';

NOTE

When the **repl load** command is used to import a database, pay attention to the following points when specifying the database name:

- If the specified database does not exist, the database will be created during the import.
- If the specified database exists and the value of **hive.repl.ckpt.key** of the database is the same as the imported path, skip the import operation.
- If the specified database already exists and no table or function exists in this database, only the tables in the source database are imported to the current database during the import. Otherwise, the import fails.

----End

12.10.4 Importing and Exporting Table/Partition Data in Hive

Scenario

In big data application scenarios, data tables in Hive usually need to be migrated to another cluster. You can run the Hive **import** and **export** commands to migrate data in tables. That is, you can run the **export** command to export Hive tables from the source cluster to the HDFS of the target cluster, run the **import** command in the target cluster to import the exported data to the corresponding Hive table.

NOTE

This section applies to MRS 3.2.0 or later.

The Hive table import and export function does not support importing or exporting encrypted tables, HBase external tables, Hudi tables, view tables, and materialized view tables.

Prerequisites

- If Hive tables or partition data is imported or exported across clusters and Kerberos authentication is enabled for both the source and destination clusters, configure cross-cluster mutual trust.
- If you want to run the **import** or **export** command to import or export tables or partitions created by other users, grant the corresponding table permission to the users.
  - If Ranger authentication is not enabled for the cluster, log in to FusionInsight Manager to grant the **Select Authorization** permission of the table corresponding to the role to which the user belongs. For details, see section **Granting Hive Permissions on Tables, Columns, or Databases**.
  - If Ranger authentication is enabled for the cluster, grant users the permission to import and export tables. For details, see **Adding a Ranger Access Permission Policy for Hive**.
- Enable the inter-cluster copy function in the source cluster and destination cluster.
- Configure the HDFS service address parameter for the source cluster to access the destination cluster.

  Log in to FusionInsight Manager of the source cluster, click **Cluster**, choose **Services** > **Hive**, and click **Configuration**. On the displayed page, search for **hdfs.site.customized.configs**, add custom parameter **dfs.namenode.rpc-address.haclusterX**, and set its value to *Service IP address of the active NameNode instance node in the destination cluster:RPC port*. Add custom parameter **dfs.namenode.rpc-address.haclusterX1** and set its value to *Service IP address of the standby NameNode instance node in the destination cluster:RPC port*. The RPC port of NameNode is **25000** by default. After saving the configuration, roll-restart the Hive service.

Procedure

Step 1 Log in to the node where the client is installed in the destination cluster as the Hive client installation user.

Step 2 Run the following command to switch to the client installation directory, for example, **/opt/client**:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 If Kerberos authentication is enabled for the cluster, run the following command to authenticate the user. Otherwise, skip this step.

kinit *Hive service user*

Step 5 Run the following command to log in to the Hive client in the destination cluster:

beeline

Step 6 Run the following command to create the **export_test** table:

create table *export_test(id int)* **;**

Step 7 Run the following command to insert data to the **export_test** table:

insert into *export_test values(123)***;**

Step 8 Repeat **Step 1** to **Step 4** in the destination cluster and run the following command to create an HDFS path for storing the exported **export_test** table:

dfs -mkdir */tmp/export*

Step 9 Run the following command to log in to the Hive client:

beeline

Step 10 Import and export the **export_test** table.

The Hive **import** and **export** commands can be used to migrate table data in the following modes. Select a proper data migration mode as required.

- Mode 1: Table export and import
  a. Run the following command in the source cluster to export the metadata and service data of the **export_test** table to the directory created in **Step 8**:

     export table *export_test* **to 'hdfs://hacluster***X/tmp/export***';**

  b. Run the following command in the destination cluster to import the table data exported in **Step 10.a** to the **export_test** table:

     import from '*/tmp/export***';**
- Mode 2: Renaming a table during the import
  a. Run the following command in the source cluster to export the metadata and service data of the **export_test** table to the directory created in **Step 8**:

     export table *export_test* **to 'hdfs://hacluster***X/tmp/export***';**

  b. Run the following command in the destination cluster to import the table data exported in **Step 10.a** to the **import_test** table:

     import table *import_test* **from '***/tmp/export***';**
- Mode 3: Partition export and import
  a. Run the following commands in the source cluster to export the **pt1** and **pt2** partitions of the **export_test** table to the directory created in **Step 8**:

     export table *export_test* **partition (***pt1***="***in***", *pt2*="***ka***") to 'hdfs://hacluster*X*/tmp/export';**

  b. Run the following command in the destination cluster to import the table data exported in **Step 10.a** to the **export_test** table:

     import from '*/tmp/export***';**

- Mode 4: Exporting table data to a Partition
  a. Run the following command in the source cluster to export the metadata and service data of the **export_test** table to the directory created in **Step 8**:

     export table *export_test* **to 'hdfs://hacluster***X/tmp/export***';**

  b. Run the following command in the destination cluster to import the table data exported in **Step 10.a** to the **pt1** and **pt2** partitions of the **import_test** table:

     import table *import_test* **partition (***pt1***="***us***",** *pt2***="***tn***") from '***/tmp/export***';**

- Mode 5: Specifying the table location during the import
  a. Run the following command in the source cluster to export the metadata and service data of the **export_test** table to the directory created in **Step 8**:

     export table *export_test* **to 'hdfs://hacluster***X/tmp/export***';**

  b. Run the following command in the destination cluster to import the table data exported in **Step 10.a** to the **import_test** table and specify its location as **tmp/export**:

     import table *import_test* **from '***/tmp***' location '***/tmp/export***';**

- Mode 6: Exporting data to an external table
  a. Run the following command in the source cluster to export the metadata and service data of the **export_test** table to the directory created in **Step 8**:

     export table *export_test* **to 'hdfs://hacluster***X/tmp/export***';**

  b. Run the following command in the destination cluster to import the table data exported in **Step 10.a** to external table **import_test**:

     import external table *import_test* **from '***/tmp/export***';**

NOTE

Before exporting table or partition data, ensure that the HDFS path for storage has been created and is empty. Otherwise, the export fails.

When partitions are exported or imported, the exported or imported table must be a partitioned table, and data of multiple partition values of the same partition field cannot be exported.

During the data import:

- If the **import from** '*/tmp/export*'**;** statement is used to import a table, the table name is not specified, and the imported data is saved to the table path with the same name as the source table. Pay attention to the following points:
  - If there is no table with the same name as that in the source cluster in the destination cluster, such a table will be created during the table import.
  - Otherwise, the HDFS directory of the table must be empty, or the import fails.
- If the **import external table** *import_test* **from** '*/tmp/export*'**;** statement is used to import a table, the exported table is imported to the specified table. Pay attention to the following points:
  - If there is no table with the same name as the specified table exists in the destination cluster, such a table will be created during the table import.
  - Otherwise, the HDFS directory of the table must be empty, or the import fails.

hacluster *X* is the value of **haclusterX** in new custom parameter**dfs.namenode.rpc-address.haclusterX**.

----End

12.10.5 Locating Abnormal Hive Files

Scenario

Data files stored in Hive are abnormal due to misoperations or disk damage, thereby causing task execution failures or incorrect data results.

Common non-text data files can be located using the specified tool.

NOTE

This section applies only to MRS 3.2.0 or later.

Procedure

1. Log in to the node where the Hive service is installed as user **omm** and run the following command to go to the Hive installation directory:

   cd ${BIGDATA_HOME}/FusionInsight_HD_*/install/FusionInsight-Hive-*/hive-*/bin

2. Run the following tool to locate abnormal Hive files:

   sh hive_parser_file.sh [--help] *<filetype> <command> <input-file|input-directory>*

   Table 12-14 describes the related parameters.

   Note: You can run only one command at a time.

Table 12-14 Parameter description

Parameter	Description	Remarks
filetype	Specifies the format of the data file to be parsed. Currently, only the ORC, RC (RCFile), and Parquet formats are supported.	Currently, data files in the RC format can only be viewed.
-c	Prints the column information in the current metadata.	The column information includes the class name, file format, and sequence number.
-d	Prints data in a data file. You can limit the data volume using the **limit** parameter.	The data is the content of the specified data file. Note that only one value can be specified for the **limit** parameter at a time.
-t	Prints the time zone to which the data is written.	The time zone is the zone to which the file is written.
-h	Prints the help information.	Help information.
-m	Prints information about various storage formats.	The information varies based on the storage format. For example, if the file format is ORC, information such as strip and block size will be printed.
-a	Prints detailed information.	The detailed information, including the preceding parameters, is displayed.
input-file	Specifies the data files to be input.	If the input directory contains a file of the supported formats, the file will be parsed. Otherwise, this operation is omitted. You can specify a local file or an HDFS/OBS file or directory.
input-directory	Specifies the directory where the input data file is located. This parameter is used when there are multiple subfiles.	

3. For example, run the following command to check the abnormal data in the **hdfs://hacluster/user/hive/warehouse/orc_test file**:

   sh hive_parser_file.sh orc -d limit=100 hdfs://hacluster/user/hive/warehouse/orc_test

   If the file name does not contain a prefix similar to **hdfs://hacluster**, the local file is read by default.

12.11 Common Hive SQL Syntax

12.11.1 Extended Hive SQL Syntax

Hive SQL supports all features of Hive-3.1.0. For details, see **https://cwiki.apache.org/confluence/display/hive/languagemanual**.

Table 12-15 describes the extended Hive statements provided by .

Table 12-15 Extended Hive statements

Extended Syntax	Syntax Description	Syntax Example	Example Description
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name (col_name data_type [COMMENT col_comment], ...) [ROW FORMAT row_format] [STORED AS file_format] | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)] ..... [TBLPROPERTIES ("groupId"=" group1 ","locatorId"="locator1")] ...;	The statement is used to create a Hive table and specify locators on which table data files locate. For details, see **Using HDFS Colocation to Store Hive Tables**.	CREATE TABLE tab1 (id INT, name STRING) row format delimited fields terminated by '\t' stored as RCFILE TBLPROPERTIES(" groupId"=" group1 ","locatorId"="locator1");	The statement is used to create table **tab1** and specify locator1 on which the table data of **tab1** locates.

Extended Syntax	Syntax Description	Syntax Example	Example Description
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name (col_name data_type [COMMENT col_comment], ...) [ROW FORMAT row_format] [STORED AS file_format] | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)] ... [TBLPROPERTIES ('**column.encode.columns**'='col_name1,col_name2'| '**column.encode.indices**'='col_id1,col_id2', **'column.encode.classname'**='encode_classname')]...;	The statement is used to create a hive table and specify the table encryption column and encryption algorithm. For details, see **Using the Hive Column Encryption**.	create table encode_test(id INT, name STRING, phone STRING, address STRING) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe' WITH SERDEPROPERTIES ('column.encode.indices'='2,3', 'column.encode.classname'='org.apache.hadoop.hive.serde2.SMS4Rewriter') STORED AS TEXTFILE;	The statement is used to create table **encode_test** and specify that column 2 and column 3 will be encrypted using the **org.apache.hadoop.hive.serde2.SMS4Rewriter** encryption algorithm class during data insertion.
REMOVE TABLE *hbase_tablename* [WHERE where_condition];	The statement is used to delete data that meets criteria from the Hive on HBase table. For details, see **Deleting Single-row Records from Hive on HBase**.	remove table hbase_table1 where id = 1;	The statement is used to delete data that meets the criterion of "id = 1" from the table.

Extended Syntax	Syntax Description	Syntax Example	Example Description
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name (col_name data_type [COMMENT col_comment], ...) [ROW FORMAT row_format] **STORED AS inputformat 'org.apache.hadoop.hive.contrib.fileformat.SpecifiedDelimiterInputFormat'** outputformat 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat';	The statement is used to create a hive table and specify that the table supports customized row delimiters. For details, see **Customizing Row Separators**.	create table blu(time string, num string, msg string) row format delimited fields terminated by ',' **stored as inputformat 'org.apache.hadoop.hive.contrib.fileformat.SpecifiedDelimiterInputFormat'** outputformat 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat';	The statement is used to create table **blu** and set **inputformat** to **SpecifiedDelimiterInputFormat** so that the query row delimiter can be specified during the query.

12.11.2 Customizing Row Separators in Hive Tables

Scenario

In most cases, a carriage return character is used as the row delimiter in Hive tables stored in text files, that is, the carriage return character is used as the terminator of a row during queries. However, some data files are delimited by special characters, and not a carriage return character.

MRS Hive allows you to use different characters or character combinations to delimit rows of Hive text data. When creating a table, set **inputformat** to **SpecifiedDelimiterInputFormat**, and set the following parameter before search each time. Then the table data is queried by the specified delimiter.

set hive.textinput.record.delimiter='';

NOTE

The Hue component of the current version does not support the configuration of multiple separators when files are imported to a Hive table.

Procedure

Step 1 Log in to the node where the Hive client is installed as the Hive client installation user.

Step 2 Run the following commands to switch to the client installation directory, configure environment variables, and authenticate users:

cd *Client installation directory*

source bigdata_env

kinit *Hive service user* (Skip this step if Kerberos authentication is not enabled for the cluster.)

Step 3 Run the following command to log in to the Hive client:

beeline

Step 4 Specify **inputFormat** and **outputFormat** when creating a table.

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] *[db_name.]table_name* **[(***col_name data_type* **[COMMENT** *col_comment***], ...)] [ROW FORMAT** *row_format***] STORED AS inputformat 'org.apache.hadoop.hive.contrib.fileformat.SpecifiedDelimiterInputFormat' outputformat 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat';**

Step 5 Specify the delimiter before search.

set hive.textinput.record.delimiter='!@!';

Hive will use '!@!' as the row delimiter.

----End

12.11.3 Syntax of Traditional Relational Databases Supported by Hive

Hive supports the following types of traditional relational database syntax:

- Grouping
- EXCEPT and INTERSECT

Grouping

Syntax description:

- Grouping takes effect only when the Group by statement contains ROLLUP or CUBE.
- The result set generated by CUBE contains all the combinations of values in the selected columns.
- The result set generated by ROLLUP contains the combinations of a certain layer structure in the selected columns.
- Grouping: If a row is added by using the CUBE or ROLLUP operator, the output value of the added row is 1. If the row is not added by using the CUBE or ROLLUP operator, the output value of the added row is 0.

For example, the **table_test** table exists in Hive and the table structure is as follows:

```
+----------------+-------------------+--+
| table_test.id  | table_test.value  |
+----------------+-------------------+--+
| 1              | 10                |
| 1              | 15                |
| 2              | 20                |
| 2              | 5                 |
| 2              | 13                |
+----------------+-------------------+--+
```

Run the following statement:

select id,grouping(id),sum(value) from table_test group by id with rollup;

The result is as follows:

```
+-------+-----------------+------+--+
|  id   | groupingresult  | sum  |
+-------+-----------------+------+--+
| 1     | 0               | 25   |
| NULL  | 1               | 63   |
| 2     | 0               | 38   |
+-------+-----------------+------+--+
```

EXCEPT and INTERSECT

- EXCEPT returns the difference of two result sets (that is, non-duplicated values return only one query).
- INTERSECT returns the intersection of two result sets (that is, non-duplicated values return by both queries).

For example, two tables **test_table1** and **test_table2** exist in Hive.

The table structure of **test_table1** is as follows:

```
+-----------------+--+
| test_table1.id  |
+-----------------+--+
| 1               |
| 2               |
| 3               |
| 4               |
+-----------------+--+
```

The table structure of **test_table2** is as follows:

```
+-----------------+--+
| test_table2.id  |
+-----------------+--+
| 2               |
| 3               |
| 4               |
| 5               |
+-----------------+--+
```

- Run the following EXCEPT statement:

  select id from test_table1 except select id from test_table2;

  The result is as follows:

  ```
  +--------------+--+
  | _alias_0.id  |
  +--------------+--+
  | 1            |
  +--------------+--+
  ```


- Run the following INTERSECT statement:

  select id from test_table1 intersect select id from test_table2;

  The result is as follows:

```
+--------------+--+
| _alias_0.id  |
+--------------+--+
| 2            |
| 3            |
| 4            |
+--------------+--+
```

12.12 Common Issues About Hive

12.12.1 How Do I Delete UDFs on Multiple HiveServers?

Question

How can I delete permanent user-defined functions (UDFs) on multiple HiveServers at the same time?

Answer

Multiple HiveServers share one MetaStore database. Therefore, there is a delay in the data synchronization between the MetaStore database and the HiveServer memory. If a permanent UDF is deleted from one HiveServer, the operation result cannot be synchronized to the other HiveServers promptly.

In this case, you need to log in to the Hive client to connect to each HiveServer and delete permanent UDFs on the HiveServers one by one. The operations are as follows:

Step 1 Log in to the node where the Hive client is installed as the Hive client installation user.

Step 2 Run the following command to go to the client installation directory:

cd *Client installation directory*

For example, if the client installation directory is **/opt/client**, run the following command:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Run the following command to authenticate the user:

kinit *Hive service user*

NOTE

The login user must have the Hive admin rights.

Step 5 Run the following command to connect to the specified HiveServer:

beeline -u "jdbc:hive2:// *10.39.151.74*:*21066*/**default;sasl.qop=auth-conf;auth=KERBEROS;principal=***hive/hadoop.<system domain name>@<system domain name>*"

NOTE

- *10.39.151.74* is the IP address of the node where the HiveServer is located.
- *21066* is the port number of the HiveServer. The HiveServer port number ranges from 21066 to 21070 by default. Use the actual port number.
- *hive* is the username. For example, if the Hive1 instance is used, the username is **hive1**.
- You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and view the value of **Local Domain**, which is the current system domain name.
- **hive/hadoop.*<system domain name>*** is the username. All letters in the system domain name contained in the username are lowercase letters.

Step 6 Run the following command to enable the Hive admin rights:

set role admin;

Step 7 Run the following command to delete the permanent UDF:

drop function *function_name*;

NOTE

- *function_name* indicates the name of the permanent function.
- If the permanent UDF is created in Spark, the permanent UDF needs to be deleted from Spark and then from HiveServer by running the preceding command.

Step 8 Check whether the permanent UDFs are deleted from all HiveServers.

- If yes, no further action is required.
- If no, go to **Step 5**.

----End

12.12.2 Why Cannot the DROP operation Be Performed on a Backed-up Hive Table?

Question

Why cannot the **DROP** operation be performed for a backed up Hive table?

Answer

Snapshots have been created for an HDFS directory mapping to the backed up Hive table, so the HDFS directory cannot be deleted. As a result, the Hive table cannot be deleted.

When a Hive table is being backed up, snapshots are created for the HDFS directory mapping to the table. The snapshot mechanism of HDFS has the following limitation: If snapshots have been created for an HDFS directory, the directory cannot be deleted or renamed unless the snapshots are deleted. When the **DROP** operation is performed for a Hive table (except the EXTERNAL table), the system attempts to delete the HDFS directory mapping to the table. If the

directory fails to be deleted, the system displays a message indicating that the table fails to be deleted.

If you need to delete this table, manually delete all backup tasks related to this table.

12.12.3 How to Perform Operations on Local Files with Hive User-Defined Functions

Question

How to perform operations on local files (such as reading the content of a file) with Hive user-defined functions?

Answer

By default, you can perform operations on local files with their relative paths in UDF. The following are sample codes:

```
public String evaluate(String text) {
  // some logic
  File file = new File("foo.txt");
  // some logic
  // do return here
}
```

In Hive, upload the file **foo.txt** used in UDF to HDFS, such as **hdfs://hacluster/tmp/foo.txt**. You can perform operations on the **foo.txt** file by creating UDF with the following sentences:

create function testFunc as 'some.class' using jar 'hdfs://hacluster/somejar.jar', file 'hdfs://hacluster/tmp/foo.txt';

In abnormal cases, if the value of **hive.fetch.task.conversion** is **more**, you can perform operations on local files in UDF by using absolute path instead of relative path. In addition, you must ensure that the file exists on all HiveServer nodes and NodeManager nodes and **omm** user have corresponding operation rights.

12.12.4 How Do I Forcibly Stop MapReduce Jobs Executed by Hive?

Question

How do I stop a MapReduce task manually if the task is suspended for a long time?

Answer

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > *Name of the desired cluster* > **Services** > **Yarn**.

Step 3 On the left pane, click **ResourceManager(Host name, Active)**, and log in to Yarn.

Step 4 Click the button corresponding to the task ID. On the task page that is displayed, click **Kill Application** in the upper left corner and click **OK** in the displayed dialog box to stop the task.

----End

12.12.5 Which special characters are not supported by Hive in complex field names

Question

Table creation fails because Hive complex fields' names contain special characters.

Answer

Hive does not support complex fields' names that contain special characters.

Special characters refer to characters other than uppercase and lowercase letters, digits, Chinese characters, and Portuguese characters.

When creating related fields, avoid using related special characters.

12.12.6 How Do I Monitor the Hive Table Size?

Question

How do I monitor the Hive table size?

Answer

The HDFS refined monitoring function allows you to monitor the size of a specified table directory.

Prerequisites

- The Hive and HDFS components are running properly.
- The HDFS refined monitoring function is normal.

Procedure

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **HDFS** > **Resource**.

Step 3 Click the first icon in the upper left corner of **Resource Usage (by Directory)**, as shown in the following figure.

Step 4 In the displayed sub page for configuring space monitoring, click **Add**.

Step 5 In the displayed **Add a Monitoring Directory** dialog box, set **Name** to the name or the user-defined alias of the table to be monitored and **Path** to the path of the

monitored table. Click **OK**. In the monitoring result, the horizontal coordinate indicates the time, and the vertical coordinate indicates the size of the monitored directory.

----End

12.12.7 How Do I Prevent Data Loss Caused by Misoperations of the insert overwrite Statement?

Question

How do I prevent key directories from data loss caused by misoperations of the **insert overwrite** statement?

Answer

During monitoring of key Hive databases, tables, or directories, to prevent data loss caused by misoperations of the **insert overwrite** statement, configure **hive.local.dir.confblacklist** in Hive to protect directories.

This configuration item has been configured for directories such as **/opt/** and **/user/hive/warehouse** by default.

Prerequisites

The Hive and HDFS components are running properly.

Procedure

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **Configurations** > **All Configurations**, and search for the **hive.local.dir.confblacklist** configuration item.

Step 3 Add paths of databases, tables, or directories to be protected in the parameter value.

Step 4 Click **Save** to save the settings.

----End

12.12.8 Why Is Hive on Spark Task Freezing When HBase Is Not Installed?

Scenario

This function applies to Hive.

Perform the following operations to configure parameters. When Hive on Spark tasks are executed in the environment where the HBase is not installed, freezing of tasks can be prevented.

NOTE

The Spark kernel version of Hive on Spark tasks has been upgraded to Spark2x. Hive on Spark tasks can be executed is Spark2x is not installed. If HBase is not installed, when Spark tasks are executed, the system attempts to connect to the ZooKeeper to access HBase until timeout occurs by default. As a result, task freezing occurs.

If HBase is not installed, perform the following operations to execute Hive on Spark tasks. If HBase is upgraded from an earlier version, you do not need to configure parameters after the upgrade.

Procedure

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **Hive** > **Configurations** > **All Configurations**.

Step 3 Choose **HiveServer(Role)** > **Customization**. Add a customized parameter to the **spark-defaults.conf** parameter file. Set **Name** to **spark.security.credentials.hbase.enabled**, and set **Value** to **false**.

Step 4 Click **Save**. In the dialog box that is displayed, click **OK**.

Step 5 Choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **Instance**, select all Hive instances, choose **More** > **Restart Instance**, enter the password, and click **OK**.

----End

12.12.9 Error Reported When the WHERE Condition Is Used to Query Tables with Excessive Partitions in FusionInsight Hive

Question

When a table with more than 32,000 partitions is created in Hive, an exception occurs during the query with the WHERE partition.

In addition, the exception information printed in **metastore.log** contains the following information:

```
Caused by: java.io.IOException: Tried to send an out-of-range integer as a 2-byte value: 32970
    at org.postgresql.core.PGStream.SendInteger2(PGStream.java:199)
    at org.postgresql.core.v3.QueryExecutorImpl.sendParse(QueryExecutorImpl.java:1330)
    at org.postgresql.core.v3.QueryExecutorImpl.sendOneQuery(QueryExecutorImpl.java:1601)
    at org.postgresql.core.v3.QueryExecutorImpl.sendParse(QueryExecutorImpl.java:1191)
    at org.postgresql.core.v3.QueryExecutorImpl.execute(QueryExecutorImpl.java:346)
```

Answer

During a query with partition conditions, HiveServer optimizes the partitions to avoid full table scanning.

All partitions whose metadata meets the conditions need to be queried.

However, the **sendOneQuery** interface provided by GaussDB limits the parameter value to **32767** in the **sendParse** method.

If the number of partition conditions exceeds **32767**, an exception occurs.

12.12.10 Why Cannot I Connect to HiveServer When I Use IBM JDK to Access the Beeline Client?

Scenario

When users check the JDK version used by the client, if the JDK version is IBM JDK, the Beeline client needs to be reconstructed. Otherwise, the client will fail to connect to HiveServer.

Procedure

Step 1 Log in to FusionInsight Manager and choose **System** > **Permission** > **User**. In the **Operation** column of the target user, choose **More** > **Download Authentication Credential**, select the cluster information, and click **OK** to download the keytab file.

Step 2 Decompress the keytab file and use WinSCP to upload the decompressed **user.keytab** file to the Hive client installation directory on the node to be operated, for example, **/opt/client**.

Step 3 Run the following command to open the **Hive/component_env** configuration file in the Hive client directory:

vi *Hive client installation directory*/**Hive/component_env**

Add the following content to the end of the line where **export CLIENT_HIVE_URI** is located:

```
\; user.principal=Username @HADOOP.COM\;user.keytab=user.keytab file path/user.keytab
```

----End

12.12.11 Description of Hive Table Location (Either Be an OBS or HDFS Path)

Question

Can Hive tables be stored in OBS or HDFS?

Answer

1. The location of a common Hive table stored on OBS can be set to an HDFS path.
2. In the same Hive service, you can create tables stored in OBS and HDFS, respectively.
3. For a Hive partitioned table stored on OBS, the location of the partition cannot be set to an HDFS path. (For a partitioned table stored on HDFS, the location of the partition cannot be changed to OBS.)

12.12.12 Why Cannot Data Be Queried After the MapReduce Engine Is Switched After the Tez Engine Is Used to Execute Union-related Statements?

Question

Hive uses the Tez engine to execute union-related statements to write data. After Hive is switched to the MapReduce engine for query, no data is found.

Answer

When Hive uses the Tez engine to execute the union-related statement, the generated output file is stored in the **HIVE_UNION_SUBDIR** directory. After Hive is switched back to the MapReduce engine, files in the directory are not read by default. Therefore, data in the **HIVE_UNION_SUBDIR** directory is not read.

In this case, you can set **mapreduce.input.fileinputformat.input.dir.recursive** to **true** to enable union optimization and determine whether to read data in the directory.

12.12.13 Why Does Hive Not Support Concurrent Data Writing to the Same Table or Partition?

Question

Why Does Data Inconsistency Occur When Data Is Concurrently Written to a Hive Table Through an API?

NOTE

This section applies only to MRS 3.1.2.

Answer

Hive does not support concurrent data insertion for the same table or partition. As a result, multiple tasks perform operations on the same temporary data directory, and one task moves the data of another task, causing task data exception. The service logic is modified so that data is inserted to the same table or partition in single thread mode.

12.12.14 Does Hive Support Vectorized Query?

Question

When the vectorized parameter **hive.vectorized.execution.enabled** is set to **true**, why do some null pointers or type conversion exceptions occur occasionally when Hive on Tez/MapReduce/Spark is executed?

Answer

Currently, Hive does not support vectorized execution.

Many community issues are introduced during vectorized execution and are not resolved stably. The default value of **hive.vectorized.execution.enabled** is **false**. You are advised not to set this parameter to **true**.

12.12.15 Why Does Metadata Still Exist When the HDFS Data Directory of the Hive Table Is Deleted by Mistake?

Question

The HDFS data directory of the Hive table is deleted by mistake, but the metadata still exists. As a result, an error is reported during task execution.

Answer

This is a exception caused by misoperation. You need to manually delete the metadata of the corresponding table and try again.

Example:

Run the following command to go to the console:

source ${BIGDATA_HOME}/FusionInsight_BASE_*xxx*/install/FusionInsight-dbservice-2.7.0/.dbservice_profile

gsql -p 20051 -U hive -d hivemeta -W HiveUser@

Run the **delete from tbls where tbl_id='xxx';** command.

12.12.16 How Do I Disable the Logging Function of Hive?

Question

How do I disable the logging function of Hive?

Answer

Step 1 Log in to the node where the client is installed as user **root**.

Step 2 Run the following command to switch to the client installation directory, for example, **/opt/client**:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Log in to the Hive client based on the cluster authentication mode.

- In security mode, run the following command to complete user authentication and log in to the Hive client:

  kinit *Component service user*

  beeline

- In normal mode, run the following command to log in to the Hive client:

- Run the following command to log in to the Hive client as the component service user:

  beeline -n *component service user*

- If no component service user is specified, the current OS user is used to log in to the Hive client.

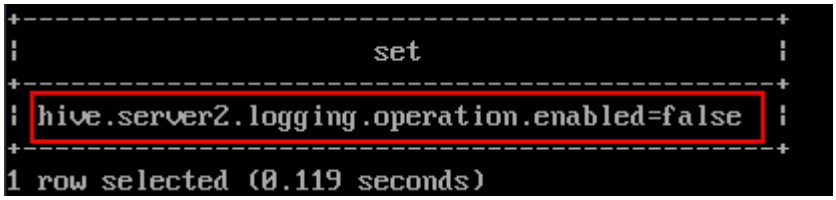
  beeline

Step 5 Run the following command to disable the logging function:

set hive.server2.logging.operation.enabled=false;

Step 6 Run the following command to check whether the logging function is disabled. If the following information is displayed, the logging function is disabled successfully.

set hive.server2.logging.operation.enabled;



----End

12.12.17 Why Hive Tables in the OBS Directory Fail to Be Deleted?

Question

In the scenario where the fine-grained permission is configured for multiple MRS users to access OBS, after the permission for deleting Hive tables in the OBS directory is added to the custom configuration of Hive.

Tables are deleted on the Hive client but still exist in the OBS directory.

Answer

You do not have the permission to delete directories on OBS. As a result, Hive tables cannot be deleted.

In this case, modify the custom IAM policy of the agency and configure Hive with the permission for deleting tables in the OBS directory.

12.12.18 Why Does an OBS Quickly Deleted Directory Not Take Effect After Being Added to the Customized Hive Configuration?

- The error message "java.lang.OutOfMemoryError: Java heap space." is displayed during Hive SQL execution.

  Solution:

  - For MapReduce tasks, increase the values of the following parameters:

    set mapreduce.map.memory.mb=8192;

    set mapreduce.map.java.opts=-Xmx6554M;

    set mapreduce.reduce.memory.mb=8192;

    set mapreduce.reduce.java.opts=-Xmx6554M;
  - For Tez tasks, increase the value of the following parameter:

    set hive.tez.container.size=8192;

- After a column name is changed to a new one using the Hive SQL **as** statement, the error message "Invalid table alias or column reference 'xxx'." is displayed when the original column name is used for compilation.

  Solution: Run the **set hive.cbo.enable=true;** statement.
- The error message "Unsupported SubQuery Expression 'xxx': Only SubQuery expressions that are top level conjuncts are allowed." is displayed during Hive SQL subquery compilation.

  Solution: Run the **set hive.cbo.enable=true;** statement.
- The error message "CalciteSubquerySemanticException [Error 10249]: Unsupported SubQuery Expression Currently SubQuery expressions are only allowed as Where and Having Clause predicates." is displayed during Hive SQL subquery compilation.

  Solution: Run the **set hive.cbo.enable=true;** statement.
- The error message "Error running query: java.lang.AssertionError: Cannot add expression of different type to set." is displayed during Hive SQL compilation.

  Solution: Run the **set hive.cbo.enable=false;** statement.
- The error message "java.lang.NullPointerException at org.apache.hadoop.hive.ql.udf.generic.GenericUDAFComputeStats $GenericUDAFNumericStatsEvaluator.init." is displayed during Hive SQL execution.

  Solution: Run the **set hive.map.aggr=false;** statement.
- When **hive.auto.convert.join** is set to **true** (enabled by default) and **hive.optimize.skewjoin** is set to **true**, the error message "ClassCastException org.apache.hadoop.hive.ql.plan.ConditionalWork cannot be cast to org.apache.hadoop.hive.ql.plan.MapredWork" is displayed.

  Solution: Run the **set hive.optimize.skewjoin=false;** statement.
- When **hive.auto.convert.join** is set to **true** (enabled by default), **hive.optimize.skewjoin** is set to **true**, and **hive.exec.parallel** is set to **true**, the error message "java.io.FileNotFoundException: File does not exist:xxx/reduce.xml" is displayed.

  Solution:
  - Method 1: Switch the execution engine to Tez. For details, see **Switching the Hive Execution Engine to Tez**.
  - Method 2: Run the **set hive.exec.parallel=false;** statement.
  - Method 3: Run the **set hive.auto.convert.join=false;** statement.
- Eerror message "NullPointerException at org.apache.hadoop.hive.ql.exec.CommonMergeJoinOperator.mergeJoinComputeKeys" is displayed when Hive on Tez executes bucket map join.

  Solution: Run the **set tez.am.container.reuse.enabled=false;** statement.

12.13 Hive Troubleshooting

12.13.1 How Do I Optimize the INSERT OVERWRITE for Reading and Writing in Same Table?

Scenario

If data needs to be inserted to the destination table using dynamic partitioning (update using historical partitions) and the destination table is the same as the data source table, running INSERT OVERWRITE on the source table may cause data loss or data inconsistency. To avoid this problem, you are advised to use a temporary table to process data, and then perform the INSERT OVERWRITE.

Procedure

The following table is taken as an example:

```
user_data(user_group int, user_name string, update_time timestamp);
```

In this table, **user_group** is the partitioning column. You need to sort the existing data by update time and update the user group information. To do so, perform the following steps:

Step 1 On the Hive Beeline CLI, enable Hive dynamic partitioning.

set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;

Step 2 Create a temporary table for storing deduplicated data.

CREATE TABLE *temp_user_data* **AS**

SELECT * FROM (

SELECT *,

ROW_NUMBER() OVER(PARTITION BY user_group ORDER BY update_time DESC) as rank

FROM user_data

) tmp

WHERE rank = 1;

Step 3 Use temporary table as the data source and insert data to the destination table.

INSERT OVERWRITE TABLE *user_data*

SELECT user_group, user_name, update_time

FROM *temp_user_data***;**

Step 4 Clear the temporary table.

DROP TABLE IF EXISTS *temp_user_data*;

----End

13 Using Hudi

13.1 Hudi Table Overview

Table Type

- Copy On Write

  Copy-on-write (COW) tables store data in Parquet files. Internal update operations need to be performed by rewriting the original Parquet files.

  - Advantage: It is efficient because only one data file in the corresponding partition needs to be read.
  - Disadvantage: During data write, a previous copy needs to be copied and then a new data file is generated based on the previous copy. This process is time-consuming. Therefore, the data read by the read request lags behind.

- Merge On Read

  Merge-on-read tables are also called MOR tables. The combination of columnar-based Parquet and row-based format Avro is used to store data. Parquet files are used to store base data, and Avro files (also called log files) are used to store incremental data.

  - Advantage: Data is written to the delta log first, and the delta log size is small. Therefore, the write cost is low.
  - Disadvantage: Files need to be compacted periodically. Otherwise, there are a large number of fragment files. The read performance is poor because delta logs and old data files need to be merged.

Hudi Table Storage

When writing data, Hudi generates a Hudi table based on attributes such as the storage path, table name, and partition structure.

Hudi table data files can be stored in the OS file system or distributed file system such as HDFS. To ensure analysis performance and data reliability, HDFS is generally used for storage. Using HDFS as an example, Hudi table storage files are classified into two types.

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS**. On the **Dashboard** tab page, click the link next to **NameNode WebUI**. On the HDFS web UI that is displayed, choose **Utilities** > **Browse the file system**.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	testcz	hadoop	0 B	Apr 25 15:32	0	0 B	.hoodie
drwxr-xr-x	testcz	hadoop	0 B	Apr 25 15:30	0	0 B	americas
drwxr-xr-x	testcz	hadoop	0 B	Apr 25 15:30	0	0 B	asia

- The **.hoodie** folder stores the log files related to file merging.

drwxr-xr-x	admintest	hadoop	0 B	Mar 30 09:44	0	0 B	.aux
drwxr-xr-x	admintest	hadoop	0 B	Mar 30 11:45	0	0 B	.temp
-rw-r--r--	admintest	hadoop	4.58 KB	Mar 30 09:44	3	128 MB	20210330094435.deltacommit
-rw-r--r--	admintest	hadoop	0 B	Mar 30 09:44	3	128 MB	20210330094435.deltacommit.inflight
-rw-r--r--	admintest	hadoop	0 B	Mar 30 09:44	3	128 MB	20210330094435.deltacommit.requested

- The path containing **_partition_key** stores actual data files and metadata by partition.

  Hudi data files of are stored in Parquet base files and Avro log files.

-rw-r--r--	admintest	hadoop	93 B	Mar 30 09:44	3	128 MB	.hoodie_partition_metadata
-rw-r--r--	admintest	hadoop	441.77 KB	Mar 30 09:46	3	128 MB	2b4d098e-4dc8-4633-a22a-dc22f87c57d9-1_0-13-22_20210330094613.parquet
-rw-r--r--	admintest	hadoop	445.28 KB	Mar 30 09:44	3	128 MB	4010e8a8-1b20-4be7-8442-4e30af401e84-0_1-4-8_20210330094435.parquet

13.2 Creating a Hudi Table Using Spark Shell

NOTE

This section applies only to MRS 3.3.1-LTS and earlier versions.

Scenario

This topic describes how to use Hudi with the Spark Shell.

The example in this topic uses the Spark data source and provides code snippets to show how to insert and update the COW table of Hudi's default storage data set and how to read snapshots and incremental data after each write operation.

Prerequisites

- The Hudi client has been downloaded and installed. Currently, the Hudi client is integrated into the Spark/Spark2x service of the MRS cluster. You can download the client that contains the Spark/Spark2x service from FusionInsight Manager. For example, the client installation directory is **/opt/hadoopclient**.
- If Kerberos authentication has been enabled for the cluster, ensure that a machine-machine user has been created on FusionInsight Manager and associated with the **hadoop** (primary group) and **hive** user groups.

Procedure

Step 1 Download and install the Hudi client. For details, see **Installing a Client**.

Step 2 Log in to the client node as the client installation user and run the following command to go to the client directory:

cd /opt/hadoopclient

Step 3 Load environment variables.

source bigdata_env

source Hudi/component_env

kinit *Created service user*

NOTE

- Change the password of the new user upon the first authentication.
- For clusters in normal mode (Kerberos authentication disabled), you do not need to run the **kinit** command.

Step 4 Run the **spark-shell --master yarn-client** command to enter the Spark Shell, import the Hudi software package, and generate test data.

- Import required packages.

  import org.apache.hudi.QuickstartUtils._

  import scala.collection.JavaConversions._

  import org.apache.spark.sql.SaveMode._

  import org.apache.hudi.DataSourceReadOptions._

  import org.apache.hudi.DataSourceWriteOptions._

  import org.apache.hudi.config.HoodieWriteConfig._

- Define the table name and storage path to generate test data.

  val tableName = "hudi_cow_table"

  val basePath = "hdfs://hacluster/tmp/hudi_cow_table"

  val dataGen = new DataGenerator

  val inserts = convertToStringList(dataGen.generateInserts(10))

  val df = spark.read.json(spark.sparkContext.parallelize(inserts, 2))

Step 5 Run the following command to write data to the Hudi table in OVERWRITE mode:

df.write.format("org.apache.hudi").

options(getQuickstartWriteConfigs).

option(PRECOMBINE_FIELD_OPT_KEY, "ts").

option(RECORDKEY_FIELD_OPT_KEY, "uuid").

option(PARTITIONPATH_FIELD_OPT_KEY, "partitionpath").

option(TABLE_NAME, tableName).

mode(Overwrite).

save(basePath)

Step 6 Register and query the temporary table.

```
val roViewDF = spark.read.format("org.apache.hudi").load(basePath + "/*/*/*/*")

roViewDF.createOrReplaceTempView("hudi_ro_table")

spark.sql("select fare, begin_lon, begin_lat, ts from hudi_ro_table where fare > 20.0").show()
```

Step 7 Generate new data and update the Hudi table in APPEND mode.

```
val updates = convertToStringList(dataGen.generateUpdates(10))

val df = spark.read.json(spark.sparkContext.parallelize(updates, 1))

df.write.format("org.apache.hudi").

options(getQuickstartWriteConfigs).

option(PRECOMBINE_FIELD_OPT_KEY, "ts").

option(RECORDKEY_FIELD_OPT_KEY, "uuid").

option(PARTITIONPATH_FIELD_OPT_KEY, "partitionpath").

option(TABLE_NAME, tableName).

mode(Append).

save(basePath)
```

Step 8 Query incremental data in the Hudi table.

- Reloading data

  ```
  spark.read.format("org.apache.hudi").load(basePath + "/*/*/*/*").createOrReplaceTempView("hudi_ro_table")
  ```

- Perform an incremental query.

  ```
  val commits = spark.sql("select distinct(_hoodie_commit_time) as commitTime from hudi_ro_table order by commitTime").map(k => k.getString(0)).take(50)

  val beginTime = commits(commits.length - 2)

  val incViewDF = spark.read.format("org.apache.hudi").

  option(VIEW_TYPE_OPT_KEY, VIEW_TYPE_INCREMENTAL_OPT_VAL).

  option(BEGIN_INSTANTTIME_OPT_KEY, beginTime).

  load(basePath);

  incViewDF.registerTempTable("hudi_incr_table")

  spark.sql("select `_hoodie_commit_time`, fare, begin_lon, begin_lat, ts from hudi_incr_table where fare > 20.0").show()
  ```

Step 9 Perform the point-in-time query.

```
val beginTime = "000"

val endTime = commits(commits.length - 2)

val incViewDF = spark.read.format("org.apache.hudi").
```


option(VIEW_TYPE_OPT_KEY, VIEW_TYPE_INCREMENTAL_OPT_VAL).

option(BEGIN_INSTANTTIME_OPT_KEY, beginTime).

option(END_INSTANTTIME_OPT_KEY, endTime).

load(basePath);

incViewDF.registerTempTable("hudi_incr_table")

spark.sql("select `_hoodie_commit_time`, fare, begin_lon, begin_lat, ts from hudi_incr_table where fare > 20.0").show()

Step 10 Delete the test data.

- Prepare the data to delete.

  val df = spark.sql("select uuid, partitionpath from hudi_ro_table limit 2")

  val deletes = dataGen.generateDeletes(df.collectAsList())

- Delete the data.

  val df = spark.read.json(spark.sparkContext.parallelize(deletes, 2));

  df.write.format("org.apache.hudi").

  options(getQuickstartWriteConfigs).

  option(OPERATION_OPT_KEY,"delete").

  option(PRECOMBINE_FIELD_OPT_KEY, "ts").

  option(RECORDKEY_FIELD_OPT_KEY, "uuid").

  option(PARTITIONPATH_FIELD_OPT_KEY, "partitionpath").

  option(TABLE_NAME, tableName).

  mode(Append).

  save(basePath);

- Query data again.

  val roViewDFAfterDelete = spark.read.format("org.apache.hudi").

  load(basePath + "/*/*/*/*")

  roViewDFAfterDelete.createOrReplaceTempView("hudi_ro_table")

  spark.sql("select uuid, partitionPath from hudi_ro_table").show()

----End

13.3 Operating a Hudi Table Using spark-sql

NOTE

This section applies only to MRS 3.5.0-LTS and later versions.

Scenario

This section describes how to use the Hudi function using spark-sql.

Prerequisites

You have created a user and added the user to user groups **hadoop** (primary group) and **hive** on Manager.

Procedure

Step 1 Download and install the Hudi client. For details, see **Installing a Client**.

NOTE

Currently, Hudi is integrated in Spark. You only need to download the Spark client on Manager. For example, the client installation directory is **/opt/client**.

Step 2 Log in to the node where the client is installed as user **root** and run the following command:

cd /opt/client

Step 3 Run the following commands to load environment variables:

source bigdata_env

source Hudi/component_env

kinit *Created user*

NOTE

- You need to change the password of the created user, and then run the **kinit** command to log in to the system again.
- In normal mode (Kerberos authentication disabled), you do not need to run the **kinit** command.
- If multiple services are installed, run the **component_env** command of the source Spark and then the **component_env** command of the source Hudi after you run the **source bigdata_env** command.

Step 4 Start spark-sql.

- Create a Hudi table.

  create table if not exists hudi_table2 (id int,name string,price double) using hudi options (type = 'cow',primaryKey = 'id',preCombineField = 'price');

- Insert data.

  insert into hudi_table2 select 1,1,1;

  insert into hudi_table2 select 2,1,1;

- Update data.

  update hudi_table2 set name=3 where id=1;

- Delete data.

  delete from hudi_table2 where id=2;

- Query data.

  select * from hudi_table2;

----End

13.4 Operating a Hudi Table Using hudi-cli.sh

Prerequisites

- For a cluster with Kerberos authentication enabled, a user has been created on FusionInsight Manager of the cluster and associated with user groups **hadoop** and **hive**.
- The Hudi cluster client has been downloaded and installed.

Basic Operations

1. Log in to the cluster client as user **root** and run the following commands:

   cd *{Client installation directory}*

   source bigdata_env

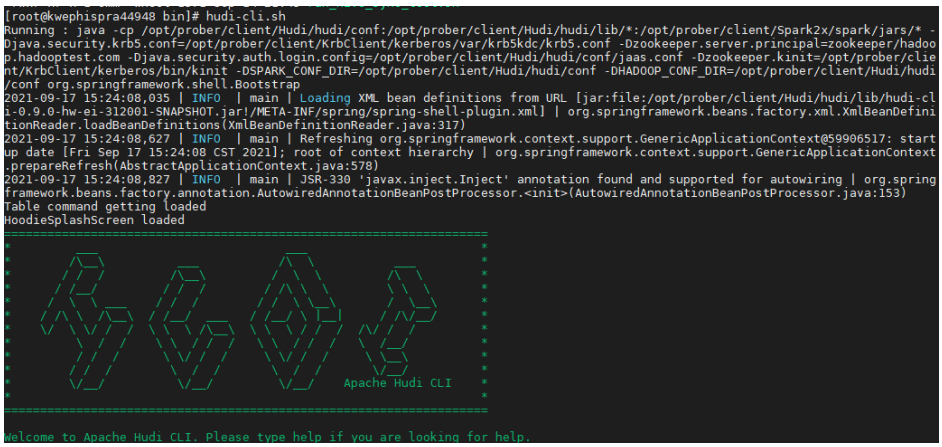
   source Hudi/component_env

   kinit *Created user*

2. Run the **hudi-cli.sh** command to access the Hudi client.

   cd *{Client installation directory}***/Hudi/hudi/bin/**

   ./hudi-cli.sh



3. Run the following example commands as required. For details about all commands, visit the Hudi official website at **https://hudi.apache.org/docs/quick-start-guide/**.

   - Viewing help information

     help // View all Hudi CLI commands.

     help 'command' // View the help information and parameter list of a certain command.

   - Connecting to a table

     connect --path '*/tmp/huditest/test_table*'

   - Viewing table information

     desc

   - Viewing compaction plans

     compactions show all

- Viewing cleaning plans

  cleans show
- Performing the cleaning operation

  cleans run
- Viewing commit information

  commits show
- Viewing the partition where the commit is written to

  commit showpartitions --commit *20210127153356*

  NOTE

  20210127153356 indicates the commit timestamp.
- Viewing the file where the commit is written to

  commit showfiles --commit *20210127153356*
- Comparing the commit information of two tables

  commits compare --path */tmp/hudimor/mytest100*
- Rolling back a commit (Only the last commit can be rolled back.)

  commit rollback --commit *20210127164905*
- Scheduling a compaction

  compaction schedule --*hoodieConfigs 'hoodie.compaction.strategy=org.apache.hudi.table.action.compact.strategy.BoundedIOCompactionStrategy,hoodie.compaction.target.io=1,hoodie.compact.inline.max.delta.commits=1'*
- Performing a compaction

  compaction run --*parallelism 100 --sparkMemory 1g --retry 1 --compactionInstant 20210602101315 --hoodieConfigs 'hoodie.compaction.strategy=org.apache.hudi.table.action.compact.strategy.BoundedIOCompactionStrategy,hoodie.compaction.target.io=1,hoodie.compact.inline.max.delta.commits=1' --propsFilePath hdfs://hacluster/tmp/default/tb_test_mor/.hoodie/hoodie.properties --schemaFilePath /tmp/default/tb_test_mor/.hoodie/compact_tb_base.json*
- Creating a savepoint

  savepoint create --commit *20210318155750*
- Rolling back a specified savepoint

  savepoint rollback --savepoint *20210318155750*

CAUTION

1. If the commit operation causes metadata conflicts, you can run the **commit rollback** and **savepoint rollback** commands to roll back data, but the Hive metadata cannot be rolled back. You can delete the Hive table and manually synchronize data.
2. The **commit rollback** command rolls back only the latest commit, and the **savepoint rollback** command rolls back only the latest savepoint. You cannot specify a commit or savepoint to roll back.

13.5 Hudi Write Operation

13.5.1 Writing Data to Hudi Tables In Batches

Scenario

Hudi provides multiple write modes. For details, see the configuration item **hoodie.datasource.write.operation**. This section describes **upsert**, **insert**, and **bulk_insert**.

- **insert**: The operation process is similar to **upsert**. The query on updated file partitions is not based on indexes. Therefore, **insert** is faster than **upsert**. This operation is recommended for data sources that do not contain updated data. If the data source contains updated data, the data lake will have duplicate data.
- **bulk_insert** (insert in batches): It is used for initial dataset loading. This operation sorts primary keys and then inserts data into a Hudi table by writing data to a common Parquet table. It has the best performance but cannot control small files. The **upsert** and **insert** operations can control small files by using heuristics.
- **upsert** (insert and update): It is the default operation type. Hudi determines whether historical data exists based on the primary key. Historical data is updated, and other data is inserted. This operation is recommended for data sources, such as change data capture (CDC), that include updated data.

NOTE

- Primary keys are not sorted during **insert**. Therefore, you are not advised to use **insert** during dataset initialization.
- You are advised to use **insert** if data is new, use **upsert** if data needs to be updated, and use **bulk_insert** if datasets need to be initialized.

Writing Data to Hudi Tables In Batches

1. Import the Hudi package to generate test data. For details, see **Step 2** to **Step 4** in **Creating a Hudi Table Using Spark Shell**.
2. Write data to the Hudi table. Add the **option("hoodie.datasource.write.operation", "bulk_insert")** parameter to the write command and set the write mode to **bulk_insert**. For details about how to specify other write modes, see **Table 13-50**.

```
df.write.format("org.apache.hudi").
options(getQuickstartWriteConfigs).
option("hoodie.datasource.write.precombine.field", "ts").
option("hoodie.datasource.write.recordkey.field", "uuid").
option("hoodie.datasource.write.partitionpath.field", "").
option("hoodie.datasource.write.operation", "bulk_insert").
option("hoodie.table.name", tableName).
option("hoodie.datasource.write.keygenerator.class",
"org.apache.hudi.keygen.NonpartitionedKeyGenerator").
option("hoodie.datasource.hive_sync.enable", "true").
option("hoodie.datasource.hive_sync.partition_fields", "").
option("hoodie.datasource.hive_sync.partition_extractor_class",
"org.apache.hudi.hive.NonPartitionedExtractor").
option("hoodie.datasource.hive_sync.table", tableName).
```



```
option("hoodie.datasource.hive_sync.use_jdbc", "false").
option("hoodie.bulkinsert.shuffle.parallelism", 4).
mode(Overwrite).
save(basePath)
```

NOTE

- For details about the parameters in the example, see **Table 13-50**.
- If the Spark DataSource API is used to update the MOR table, small files of the updated data may be merged when a small volume of data is inserted. As a result, some updated data can be found in the read-optimized view of the MOR table.
- If the base file of the data to be updated is a small file, the data to be inserted and new data for update are merged with the base file to generate a new base file instead of being written to logs.

Configuring Partitions

Hudi supports multiple partitioning modes, such as multi-level partitioning, non-partitioning, single-level partitioning, and partitioning by date. You can select a proper partitioning mode as required. The following describes how to configure different partitioning modes for Hudi.

- Multi-level partitioning

  Multi-level partitioning indicates that multiple fields are specified as partition keys. Pay attention to the following configuration items:

Configuration Item	Description
hoodie.datasource.write.partitionpath.field	Configure multiple partition fields, for example, **p1**, **p2**, and **p3**.
hoodie.datasource.hive_sync.partition_fields	Set this parameter the same as **hoodie.datasource.write.partitionpath.field**.
hoodie.datasource.write.keygenerator.class	Set this parameter to **org.apache.hudi.keygen.ComplexKeyGenerator**.
hoodie.datasource.hive_sync.partition_extractor_class	Set this parameter to **org.apache.hudi.hive.MultiPartKeysValueExtractor**.

```
df.write.format("org.apache.hudi").
options(getQuickstartWriteConfigs).
option("hoodie.datasource.write.precombine.field", "ts").
option("hoodie.datasource.write.recordkey.field", "uuid").
option("hoodie.datasource.write.partitionpath.field", "p1,p2,p3").
option("hoodie.datasource.write.operation", "bulk_insert").
option("hoodie.table.name", tableName).
option("hoodie.datasource.write.keygenerator.class",
"org.apache.hudi.keygen.ComplexKeyGenerator").
option("hoodie.datasource.hive_sync.enable", "true").
option("hoodie.datasource.hive_sync.partition_fields", "p1,p2,p3").
option("hoodie.datasource.hive_sync.partition_extractor_class",
"org.apache.hudi.hive.MultiPartKeysValueExtractor").
option("hoodie.datasource.hive_sync.table", tableName).
option("hoodie.datasource.hive_sync.use_jdbc", "false").
```



```
option("hoodie.bulkinsert.shuffle.parallelism", 4).
mode(Overwrite).
save(basePath)
```

- Non-partitioning

  Hudi supports non-partitioned tables. Pay attention to the following configuration items:

Configuration Item	Description
hoodie.datasource.write.partitionpath.field	Leave this parameter blank.
hoodie.datasource.hive_sync.partition_fields	Leave this parameter blank.
hoodie.datasource.write.keygenerator.class	Set this parameter to **org.apache.hudi.keygen.NonpartitionedKeyGenerator**.
hoodie.datasource.hive_sync.partition_extractor_class	Set this parameter to **org.apache.hudi.hive.NonPartitionedExtractor**.

```
df.write.format("org.apache.hudi").
options(getQuickstartWriteConfigs).
option("hoodie.datasource.write.precombine.field", "ts").
option("hoodie.datasource.write.recordkey.field", "uuid").
option("hoodie.datasource.write.partitionpath.field", "").
option("hoodie.datasource.write.operation", "bulk_insert").
option("hoodie.table.name", tableName).
option("hoodie.datasource.write.keygenerator.class",
"org.apache.hudi.keygen.NonpartitionedKeyGenerator").
option("hoodie.datasource.hive_sync.enable", "true").
option("hoodie.datasource.hive_sync.partition_fields", "").
option("hoodie.datasource.hive_sync.partition_extractor_class",
"org.apache.hudi.hive.NonPartitionedExtractor").
option("hoodie.datasource.hive_sync.table", tableName).
option("hoodie.datasource.hive_sync.use_jdbc", "false").
option("hoodie.bulkinsert.shuffle.parallelism", 4).
mode(Overwrite).
save(basePath)
```

- Single-level partitioning

  It is similar to multi-level partitioning. Pay attention to the following configuration items:

Configuration Item	Description
hoodie.datasource.write.partitionpath.field	Set this parameter to one field, for example, **p**.
hoodie.datasource.hive_sync.partition_fields	Set this same as **hoodie.datasource.write.partitionpath.field**.

Configuration Item	Description
hoodie.datasource.write.keygenerator.class	By default, you can set this parameter to **org.apache.hudi.keygen.SimpleKeyGenerator** and **org.apache.hudi.keygen.ComplexKeyGenerator**. You can also leave this parameter blank.
hoodie.datasource.hive_sync.partition_extractor_class	Set this parameter to **org.apache.hudi.hive.MultiPartKeysValueExtractor**.

```
df.write.format("org.apache.hudi").
options(getQuickstartWriteConfigs).
option("hoodie.datasource.write.precombine.field", "ts").
option("hoodie.datasource.write.recordkey.field", "uuid").
option("hoodie.datasource.write.partitionpath.field", "p").
option("hoodie.datasource.write.operation", "bulk_insert").
option("hoodie.table.name", tableName).
option("hoodie.datasource.write.keygenerator.class",
"org.apache.hudi.keygen.ComplexKeyGenerator").
option("hoodie.datasource.hive_sync.enable", "true").
option("hoodie.datasource.hive_sync.partition_fields", "p").
option("hoodie.datasource.hive_sync.partition_extractor_class",
"org.apache.hudi.hive.MultiPartKeysValueExtractor").
option("hoodie.datasource.hive_sync.table", tableName).
option("hoodie.datasource.hive_sync.use_jdbc", "false").
option("hoodie.bulkinsert.shuffle.parallelism", 4).
mode(Overwrite).
save(basePath)
```

- Partitioning by date

  The **date** field is specified as the partition field. Pay attention to the following configuration items:

Configuration Item	Description
hoodie.datasource.write.partitionpath.field	Set this parameter to the **date** field.
hoodie.datasource.hive_sync.partition_fields	Set this same as **hoodie.datasource.write.partitionpath.field**.
hoodie.datasource.write.keygenerator.class	Set this parameter to **org.apache.hudi.keygen.ComplexKeyGenerator**.
hoodie.datasource.hive_sync.partition_extractor_class	Set this parameter to **org.apache.hudi.hive.SlashEncodedDayPartitionValueExtractor**.

NOTE

Date format for **SlashEncodedDayPartitionValueExtractor** must be *yyyy/mm/dd*.

- Partition sorting

Configuration Item	Description
hoodie.bulkinsert.user.defined.partitioner.class	Partition sorting class. You can customize a sorting method. For details, see the sample code.

NOTE

By default, **bulk_insert** sorts data by character and applies only to primary keys of StringType.

13.5.2 Writing Data to Hudi Tables in Streams

NOTE

This section applies only to MRS 3.3.1-LTS and earlier versions.

Stream Write Using HoodieDeltaStreamer

The HoodieDeltaStreamer tool provided by Hudi supports stream write. You can also use SparkStreaming to write data in microbatch mode. HoodieDeltaStreamer provides the following functions:

- Supports multiple data sources, such as Kafka and DFS.
- Manages checkpoints, rollback, and recovery to ensure exactly-once semantics.
- Supports user-defined transformations.

Example:

Prepare the configuration file **kafka-source.properties**.

```
#Hudi configuration
hoodie.datasource.write.recordkey.field=id
hoodie.datasource.write.partitionpath.field=age
hoodie.upsert.shuffle.parallelism=100
#hive config
hoodie.datasource.hive_sync.table=hudimor_deltastreamer_partition
hoodie.datasource.hive_sync.partition_fields=age
hoodie.datasource.hive_sync.partition_extractor_class=org.apache.hudi.hive.MultiPartKeysValueExtractor
hoodie.datasource.hive_sync.use_jdbc=false
hoodie.datasource.hive_sync.support_timestamp=true
# Kafka Source topic
hoodie.deltastreamer.source.kafka.topic=hudimor_deltastreamer_partition
#checkpoint
hoodie.deltastreamer.checkpoint.provider.path=hdfs://hacluster/tmp/huditest/
hudimor_deltastreamer_partition
# Kafka props
# The kafka cluster we want to ingest from
bootstrap.servers= xx.xx.xx.xx:xx
auto.offset.reset=earliest
#auto.offset.reset=latest
```



```
group.id=hoodie-delta-streamer
offset.rang.limit=10000
```

Run the following commands to specify the HoodieDeltaStreamer execution parameters (for details about the parameter configuration, visit the official website at **https://hudi.apache.org/**):

spark-submit --master yarn

--jars /opt/hudi-java-examples-1.0.jar // Specify the Hudi **jars** directory required for Spark running.

--driver-memory 1g

--executor-memory 1g --executor-cores 1 --num-executors 2 --conf spark.kryoserializer.buffer.max=128m

--driver-class-path /opt/client/Hudi/hudi/conf:/opt/client/Hudi/hudi/lib/*:/opt/client/Spark2x/spark/jars/*:/opt/hudi-examples-0.6.1-SNAPSHOT.jar:/opt/hudi-examples-0.6.1-SNAPSHOT-tests.jar // Specify the Hudi **jars** directory required by the Spark driver.

--class org.apache.hudi.utilities.deltastreamer.HoodieDeltaStreamer spark-internal

--props file:///opt/kafka-source.properties // Specify the configuration file. You need to set the configuration file path to the HDFS path when submitting tasks in yarn-cluster mode.

--target-base-path /tmp/huditest/hudimor1_deltastreamer_partition // Specify the path of the Hudi table.

--table-type MERGE_ON_READ // Specify the type of the Hudi table to be written.

--target-table hudimor_deltastreamer_partition // Specify the Hudi table name.

--source-ordering-field name // Specify the columns to be pre-combined in the Hudi table.

--source-class org.apache.hudi.utilities.sources.JsonKafkaSource // Set the consumed data source to **JsonKafkaSource**. Specify source classes based on data sources.

--schemaprovider-class com.huaweixxx.bigdata.hudi.examples.DataSchemaProviderExample // Specify the schema required by the Hudi table.

--transformer-class com.huaweixxx.bigdata.hudi.examples.TransformerExample // Specify how to process the data obtained from the data source. Set this parameter based on service requirements.

--enable-hive-sync // Enable Hive synchronization to synchronize the Hudi table to Hive.

--continuous // Set the stream processing mode to **continuous**.

Stream Write Using HoodieMultiTableDeltaStreamer

NOTE

HoodieMultiTableDeltaStreamer streaming write is available only in MRS 3.2.0 or later.

HoodieDeltaStreamer allows you to capture data from multiple types of source tables and write the data to Hudi tables. However, you can only write data in one source table to one destination table. By contrast, HoodieMultiTableDeltaStreamer supports data write from multiple source tables to one or multiple destination tables.

- **The following example describes how to write data in two Kafka source tables to two Hudi tables.**

  NOTE

  Set the following parameters:

  ```
  // Specify the target table.
  hoodie.deltastreamer.ingestion.tablesToBeIngested=Directory name.target table
  //Specify all source tables to specific destination tables.
  hoodie.deltastreamer.source.sourcesBoundTo.Destination table=Directory name.Source table 1,Directory name.Source table 2
  // Specify the configuration file path of each source table.
  Hoodie.deltastreamer.Source.directory name.Source table 1.configFile=Path 1
  Hoodie.deltastreamer.source.Directory name.Source table 2.configFile=Path 2
  // Specify the check point of each source table. The format of the recovery point varies according to the source table type. For example, the recovery point format of Kafka source is "Topic name,Partition name:offset".
  hoodie.deltastreamer.current.source.checkpoint=Topic name,Partition name:offset
  // Specify the associated table (Hudi table) of each source table. If there are multiple associated tables, separate them with commas (,).
  hoodie.deltastreamer.source.associated.tables=hdfs://hacluster/....., hdfs://hacluster/.....
  // Specify the transform operation before the data in each source table is written to Hudi. Note that the columns to be written must be listed. Do not use select *.
  // <SRC> indicates the current source table and cannot be changed.
  hoodie.deltastreamer.transformer.sql=select field1,field2,field3,... from <SRC>
  ```

  Spark submission command:

  ```
  spark-submit \
  --master yarn \
  --driver-memory 1g \
  --executor-memory 1g \
  --executor-cores 1 \
  --num-executors 5 \
  --conf spark.driver.extraClassPath=/opt/client/Hudi/hudi/conf:/opt/client/Hudi/hudi/lib/*:/opt/client/Spark2x/spark/jars/* \
  --class org.apache.hudi.utilities.deltastreamer.HoodieMultiTableDeltaStreamer /opt/client/Hudi/hudi/lib/hudi-utilities_2.12-*.jar \
  --props file:///opt/hudi/testconf/sourceCommon.properties \
  --config-folder file:///opt/hudi/testconf/ \
  --source-class org.apache.hudi.utilities.sources.JsonKafkaSource \
  --schemaprovider-class org.apache.hudi.examples.common.HoodieMultiTableDeltaStreamerSchemaProvider \
  --transformer-class org.apache.hudi.utilities.transform.SqlQueryBasedTransformer \
  --source-ordering-field col6 \
  --base-path-prefix hdfs://hacluster/tmp/ \
  --table-type COPY_ON_WRITE \
  --target-table KafkaToHudi \
  --enable-hive-sync \
  --allow-fetch-from-multiple-sources \
  --allow-continuous-when-multiple-sources
  ```


NOTE

1. When the **source** type is **kafka source**, the schema provider class specified by **--schemaprovider-class** needs to be developed by users.
2. **--allow-fetch-from-multiple-sources** indicates that multi-source table writing is enabled.
3. **--allow-continuous-when-multiple-sources** indicates that multi-source table continuous write is enabled. If this parameter is not set, the task ends after all source tables are written once.

sourceCommon.properties:

```
hoodie.deltastreamer.ingestion.tablesToBeIngested=testdb.KafkaToHudi
hoodie.deltastreamer.source.sourcesBoundTo.KafkaToHudi=source1,source2
hoodie.deltastreamer.source.default.source1.configFile=file:///opt/hudi/testconf/source1.properties
hoodie.deltastreamer.source.default.source2.configFile=file:///opt/hudi/testconf/source2.properties

hoodie.datasource.write.keygenerator.class=org.apache.hudi.keygen.SimpleKeyGenerator
hoodie.datasource.write.partitionpath.field=col0
hoodie.datasource.write.recordkey.field=primary_key
hoodie.datasource.write.precombine.field=col6

hoodie.datasource.hive_sync.table=kafkatohudisync
hoodie.datasource.hive_sync.partition_fields=col0
hoodie.datasource.hive_sync.partition_extractor_class=org.apache.hudi.hive.MultiPartKeysValueExtracto
r

bootstrap.servers=192.168.34.221:21005,192.168.34.136:21005,192.168.34.175:21005
auto.offset.reset=latest
group.id=hoodie-test
```

source1.properties:

```
hoodie.deltastreamer.current.source.name=source1 // Specify the name of a Kafka source table.
hoodie.deltastreamer.source.kafka.topic=s1
hoodie.deltastreamer.current.source.checkpoint=s1,0:0,1:0 // Checkpoint of the source table when the
task is started. The deltastreamer tasks resume from offset 0 of partition 0 and offset 0 of partition 1.
// Specify the Hudi table to be combined with the source1 table. If the Hudi table has been
synchronized to Hive, skip this step and use the table name in the SQL statement.
hoodie.deltastreamer.source.associated.tables=hdfs://hacluster/tmp/huditest/tb_test_cow_par
// <SRC> indicates the current source table, that is, source1. The value is fixed.
hoodie.deltastreamer.transformer.sql=select A.primary_key, A.col0, B.col1, B.col2, A.col3, A.col4, B.col5,
B.col6, B.col7 from <SRC> as A join tb_test_cow_par as B on A.primary_key = B.primary_key
```

source2.properties

```
hoodie.deltastreamer.current.source.name=source2
hoodie.deltastreamer.source.kafka.topic=s2
hoodie.deltastreamer.current.source.checkpoint=s2,0:0,1:0
hoodie.deltastreamer.source.associated.tables=hdfs://hacluster/tmp/huditest/tb_test_cow_par
hoodie.deltastreamer.transformer.sql=select A.primary_key, A.col0, B.col1, B.col2, A.col3, A.col4, B.col5,
B.col6, B.col7 from <SRC> as A join tb_test_cow_par as B on A.primary_key = B.primary_key
```

- **The following example describes how to write data in two Hudi tables to one Hudi table**

  Spark submission command:

```
spark-submit \
--master yarn \
--driver-memory 1g \
--executor-memory 1g \
--executor-cores 1 \
--num-executors 2 \
--conf spark.driver.extraClassPath=/opt/client/Hudi/hudi/conf:/opt/client/Hudi/hudi/lib/*:/opt/client/
Spark2x/spark/jars/* \
--class org.apache.hudi.utilities.deltastreamer.HoodieMultiTableDeltaStreamer /opt/client/Hudi/
hudi/lib/hudi-utilities_2.12-*.jar \
--props file:///opt/testconf/sourceCommon.properties \
--config-folder file:///opt/testconf/ \
--source-class org.apache.hudi.utilities.sources.HoodieIncrSource \ // Specify that the source table is a
```



```
Hudi table, which can only be COW.
--payload-class org.apache.hudi.common.model.OverwriteNonDefaultsWithLatestAvroPayload \ // Specify a payload, which determines how the original value is changed to a new value.
--transformer-class org.apache.hudi.utilities.transform.SqlQueryBasedTransformer \ // Specify a transformer class. If the schema of the source table is different from that of the target table, the source table data can be written to the target table only after being transformed.
--source-ordering-field col6 \
--base-path-prefix hdfs://hacluster/tmp/ \ // Path for saving the destination tables
--table-type MERGE_ON_READ \ // Type of the destination table, which can be COW or MOR.
--target-table tb_test_mor_par_300 \ // Specify the name of the target table. When you write data in multiple source tables to a target table, the name of the target table must be specified.
--checkpoint 000 \ // Specify a checkpoint (commit timestamp), which indicates that Delta Streamer is restored from this checkpoint. 000 indicates that Delta Streamer is restored from the beginning.
--enable-hive-sync \
--allow-fetch-from-multiple-sources \
--allow-continuous-when-multiple-sources \
--op UPSERT // Specify the write type.
```

NOTE

- If the **source** type is **HoodieIncrSourc**, **--schemaprovider-class** does not need to be specified.
- If **transformer-class** is set to **SqlQueryBasedTransformer**, you can use SQL queries to convert the data structure of the source table to that of the destination table.

file:///opt/testconf/sourceCommon.properties:

```
# Common properties of source tables
hoodie.deltastreamer.ingestion.tablesToBeIngested=testdb.tb_test_mor_par_300 // Specify a target table (common property) to which multiple source tables are written.
hoodie.deltastreamer.source.sourcesBoundTo.tb_test_mor_par_300=testdb.tb_test_mor_par_100,testdb.tb_test_mor_par_200 //Specify multiple source tables.
hoodie.deltastreamer.source.testdb.tb_test_mor_par_100.configFile=file:///opt/testconf/tb_test_mor_par_100.properties // Property file path of the source table tb_test_mor_par_100
hoodie.deltastreamer.source.testdb.tb_test_mor_par_200.configFile=file:///opt/testconf/tb_test_mor_par_200.properties //Property file path of the source table tb_test_mor_par_200

# Hudi write configurations shared by all source tables. The independent configurations of a source table need to be written to its property file.
hoodie.datasource.write.keygenerator.class=org.apache.hudi.keygen.SimpleKeyGenerator
hoodie.datasource.write.partitionpath.field=col0
hoodie.datasource.write.recordkey.field=primary_key
hoodie.datasource.write.precombine.field=col6
```

file:///opt/testconf/tb_test_mor_par_100.properties

```
# Configurations of the source table tb_test_mor_par_100
hoodie.deltastreamer.source.hoodieincr.path=hdfs://hacluster/tmp/testdb/tb_test_mor_par_100 // Path of the source table
hoodie.deltastreamer.source.hoodieincr.partition.fields=col0 // Partitioning key of the source table
hoodie.deltastreamer.source.hoodieincr.read_latest_on_missing_ckpt=false
hoodie.deltastreamer.source.associated.tables=hdfs://hacluster/tmp/testdb/tb_test_mor_par_400 // Specify the table to be associated with the source table.
hoodie.deltastreamer.transformer.sql=select A.primary_key, A.col0, B.col1, B.col2, A.col3, A.col4, B.col5, A.col6, B.col7 from <SRC> as A join tb_test_mor_par_400 as B on A.primary_key = B.primary_key //This configuration takes effect only when transformer-class is set to SqlQueryBasedTransformer.
```

file:///opt/testconf/tb_test_mor_par_200.properties

```
# Configurations of the source table tb_test_mor_par_200
hoodie.deltastreamer.source.hoodieincr.path=hdfs://hacluster/tmp/testdb/tb_test_mor_par_200
hoodie.deltastreamer.source.hoodieincr.partition.fields=col0
hoodie.deltastreamer.source.hoodieincr.read_latest_on_missing_ckpt=false
hoodie.deltastreamer.source.associated.tables=hdfs://hacluster/tmp/testdb/tb_test_mor_par_400
hoodie.deltastreamer.transformer.sql=select A.primary_key, A.col0, B.col1, B.col2, A.col3, A.col4, B.col5, A.col6, B.col7 from <SRC> as A join tb_test_mor_par_400 as B on A.primary_key = B.primary_key // Convert the data structure of the source table to that of the destination table. If the source table needs to be associated with Hive, you can use the table name in the SQL query for association. If the source table needs to be associated with a Hudi table, you need to specify the path of the Hudi table first and then use the table name in the SQL query for association.
```


13.5.3 Synchronizing Hudi Table Data to Hive

You can run **run_hive_sync_tool.sh** to synchronize data in the Hudi table to Hive.

For example, run the following command to synchronize the Hudi table in the **hdfs://hacluster/tmp/huditest/hudimor1_deltastreamer_partition** directory on HDFS to the Hive table **table hive_sync_test3** with **unite**, **country**, and **state** as partition keys:

run_hive_sync_tool.sh --partitioned-by unite,country,state --base-path hdfs://hacluster/tmp/huditest/hudimor1_deltastreamer_partition --table hive_sync_test3 --partition-value-extractor org.apache.hudi.hive.MultiPartKeysValueExtractor --support-timestamp

Table 13-1 Parameter description

Command	Description	Mandatory or Not (Yes or No)	Default Value
--database	Specifies the Hive database name.	No	default
--table	Specifies the Hive table name.	Yes	-
--base-file-format	Specifies the file format (**PARQUET** or **HFILE**).	No	PARQUET
--user	Specifies the Hive username.	No	-
--pass	Specifies the Hive password.	No	-
--jdbc-url	Specifies the Hive JDBC connection URL.	No	-
--base-path	Specifies the storage path of the Hudi table to be synchronized.	Yes	-
--partitioned-by	Specifies the partition key.	No	-
--partition-value-extractor	Specifies the partition class. PartitionValueExtractor needs to be implemented. The partition value can be extracted from the HDFS path.	No	SlashEncodedDayPartitionValueExtractor

Command	Description	Mandatory or Not (Yes or No)	Default Value
--assume-date-partitioning	Creates partitions in yyyy/mm/dd format to support backward compatibility.	No	false
--use-pre-apache-input-format	Use InputFormat in the **com.uber.hoodie** package to replace the one in the **org.apache.hudi** package. Do not use this command except for migrating projects from **com.uber.hoodie** to **org.apache.hudi**.	No	false
--use-jdbc	Uses Hive JDBC connection.	No	true
--auto-create-database	Specifies whether to automatically create a Hive database.	No	true
--skip-ro-suffix	Specifies whether to skip the read-optimized view with the **_ro** suffix during registration.	No	false
--use-file-listing-from-metadata	Specifies whether to obtain the file list from the Hudi metadata.	No	false
--verify-metadata-file-listing	Specifies whether to verify the file list in the Hudi metadata based on the file system.	No	false
--help/-h	Specifies whether to display help information.	No	false
--support-timestamp	Specifies whether to convert **TIMESTAMP_MICROS** of INT64 to Hive timestamp.	No	false
--decode-partition	Specifies whether to decode the partition value if the partition is encoded during the write process.	No	false

Command	Description	Mandatory or Not (Yes or No)	Default Value
--batch-sync-num	Specifies the number of Hive partitions to be synchronized in each batch.	No	1000

NOTE

During Hive synchronization, if the table does not exist, an external table is created and partitions are added. If the table exists, check whether table schemas are different. If they are different, replace the table. Check whether new partitions exist. If new partitions exist, partitions are added accordingly.

Therefore, there are the following restrictions when Hive synchronization is used:

- Fields can only be added to the schema and cannot be modified or deleted.
- Partition directories can only be added but cannot be deleted.
- **Overwrite** can only overwrite the Hudi table. The Hive table cannot be overwritten synchronously.
- Do not use the timestamp type as the partition column when synchronizing a Hudi table to Hive.

13.6 Hudi Read Operation

13.6.1 Hudi Read

Read operations on Hudi tables are based on three types of views. You can select a proper view for query as required.

Hudi supports multiple query engines, including Spark, Hive, and HetuEngine. For details, see **Table 13-2** and **Table 13-3**.

Table 13-2 COW tables

Query Engine	Real-time View/Read-optimized View	Incremental View
Hive	Y	Y
Spark (SparkSQL)	Y	Y
Spark (SparkDataSource API)	Y	Y
HetuEngine	Y	N

Table 13-3 MOR tables

Query Engine	Real-time View	Incremental View	Read-optimized View
Hive	Y	Y	Y
Spark (SparkSQL)	Y	Y	Y
Spark (SparkDataSource API)	Y	Y	Y
HetuEngine	Y	N	Y

CAUTION

- Currently, the partition deduction capability is not supported when Hudi uses the Spark DataSource API to read data. For example, when the DataSource API is used to query a bootstrap table, the partition field may not be displayed or may be displayed as null.
- For an incremental view, set **hoodie.hudicow.consume.mode** to **INCREMENTAL**. This parameter applies only to queries on the incremental view and cannot be used for queries on other types of Hudi tables or queries on other tables. You can set **hoodie.hudicow.consume.mode** to **SNAPSHOT** or any value to restore the configuration.

13.6.2 Reading the Hudi COW Table View

- Reading the real-time view (using Hive and SparkSQL as an example): Directly read the Hudi table stored in Hive and use **${***table_name***}** to specify the table name.

```
select count(*) from ${table_name};
```

- Reading the real-time view (using the Spark DataSource API as an example): This is similar to reading a common DataSource table.

  The query type **QUERY_TYPE_OPT_KEY** must be set to **QUERY_TYPE_SNAPSHOT_OPT_VAL**. Use **${***table_name***}** to specify the table name.

```
spark.read.format("hudi")
.option(QUERY_TYPE_OPT_KEY, QUERY_TYPE_SNAPSHOT_OPT_VAL) // Set the query type to the real-time view.
.load("/tmp/default/cow_bugx/") // Specify the path of the Hudi table to read.
.createTempView("mycall")
spark.sql("select * from mycall").show(100)
```

- Reading the incremental view (using Hive as an example and **${***table_name***}** to specify the table name.)

```
set hoodie.${table_name}.consume.mode=INCREMENTAL;  //Set incremental read.
set hoodie.${table_name}.consume.max.commits=3;  // Specify the maximum number of commits to be consumed.
set hoodie.${table_name}.consume.start.timestamp=20201227153030;  // Specify the initial commit to pull incremental views.
select count(*) from default.${table_name} where `_hoodie_commit_time`>'20201227153030'; // This filtering condition must be added, and the value is the initial commit to pull incremental views.
```


- Reading the incremental view (using SparkSQL as an example and **${*table_name*}** to specify the table name.)

```
set hoodie.${table_name}.consume.mode=INCREMENTAL;  //Set incremental read.
set hoodie.${table_name}.consume.start.timestamp=20201227153030;  // Specify the initial commit to pull incremental views.
set hoodie.${table_name}.consume.end.timestamp=20210308212318;  // Specify the end commit to pull incremental views. If this parameter is not specified, the latest commit is used.
select count(*) from default.${table_name} where `_hoodie_commit_time`>'20201227153030'; // This filtering condition must be added, and the value is the initial commit to pull incremental views.
```

- Reading the incremental view (using the Spark DataSource API as an example):

  QUERY_TYPE_OPT_KEY must be set to **QUERY_TYPE_INCREMENTAL_OPT_VAL**.

```
spark.read.format("hudi")
.option(QUERY_TYPE_OPT_KEY, QUERY_TYPE_INCREMENTAL_OPT_VAL) // Set the query type to the incremental mode.
.option(BEGIN_INSTANTTIME_OPT_KEY, "20210308212004")  // Specify the initial incremental pull commit.
.option(END_INSTANTTIME_OPT_KEY, "20210308212318")  //: Specify the end commit of the incremental pull.
.load("/tmp/default/cow_bugx/")  // Specify the path of the Hudi table to read.
.createTempView("mycall")  // Register as a Spark temporary table.
spark.sql("select * from mycall where `_hoodie_commit_time`>'20210308211131'")// Start the query. The statement is the same as the Hive incremental query statement.
.show(100, false)
```

- Reading the read-optimized view: The read-optimized view of COW tables is equivalent to the real-time view.

13.6.3 Reading the Hudi MOR Table View

After the MOR table is synchronized to Hive, the following two tables are synchronized to Hive: *Table name*_**rt** and *Table name*_**ro**. The table suffixed with **rt** indicates the real-time view, and the table suffixed with **ro** indicates the read-optimized view. For example, if the hudi table **${*table_name*}** is synchronized to Hive, two extra tables **${*table_name*}_rt** and **${*table_name*}_ro** are generated in the Hive table after synchronization.

- Reading the real-time view (using Hive and SparkSQL as an example): Directly read the Hudi table with suffix **_rt** stored in Hive.

```
select count(*) from ${table_name}_rt;
```

- Reading the real-time view (using the Spark DataSource API as an example): The operations are the same as those for the COW table. For details, see the operations for the COW table.
- Reading the incremental view (using Hive as an example):

```
set hive.input.format=org.apache.hudi.hadoop.hive.HoodieCombineHiveInputFormat; // This parameter does not need to be specified for SparkSQL.
set hoodie.${table_name}.consume.mode=INCREMENTAL;
set hoodie.${table_name}.consume.max.commits=3;
set hoodie.${table_name}.consume.start.timestamp=20201227153030;
select count(*) from default.${table_name}_rt where `_hoodie_commit_time`>'20201227153030';
```

- Reading the incremental view (using Spark SQL as an example):

```
set hoodie.${table_name}.consume.mode=INCREMENTAL;
set hoodie.${table_name}.consume.start.timestamp=20201227153030;  // Specify the initial commit to pull incremental views.
set hoodie.${table_name}.consume.end.timestamp=20210308212318;  // Specify the end commit to pull incremental views. If this parameter is not specified, the latest commit is used.
select count(*) from default.${table_name}_rt where `_hoodie_commit_time`>'20201227153030';
```


- Incremental view (using the Spark DataSource API as an example): The operations are the same as those for the COW table. For details, see the operations for the COW table.
- Reading the read-optimized view (using Hive and SparkSQL as an example): Directly read the Hudi table with suffix **_ro** stored in Hive.

  ```
  select count(*) from ${table_name}_ro;
  ```

- Reading the read-optimized view (using the Spark DataSource API as an example): This is similar to reading a common DataSource table.

  QUERY_TYPE_OPT_KEY must be set to **QUERY_TYPE_READ_OPTIMIZED_OPT_VAL**.

  ```
  spark.read.format("hudi")
  .option(QUERY_TYPE_OPT_KEY, QUERY_TYPE_READ_OPTIMIZED_OPT_VAL) // Set the query type to the read-optimized view.
  .load("/tmp/default/mor_bugx/") // Specify the path of the Hudi table to read.
  .createTempView("mycall")
  spark.sql("select * from mycall").show(100)
  ```

13.7 Hudi Data Management and Maintenance

13.7.1 Hudi Clustering

Introduction

Clustering reorganizes data layout to improve query performance without affecting the ingestion speed.

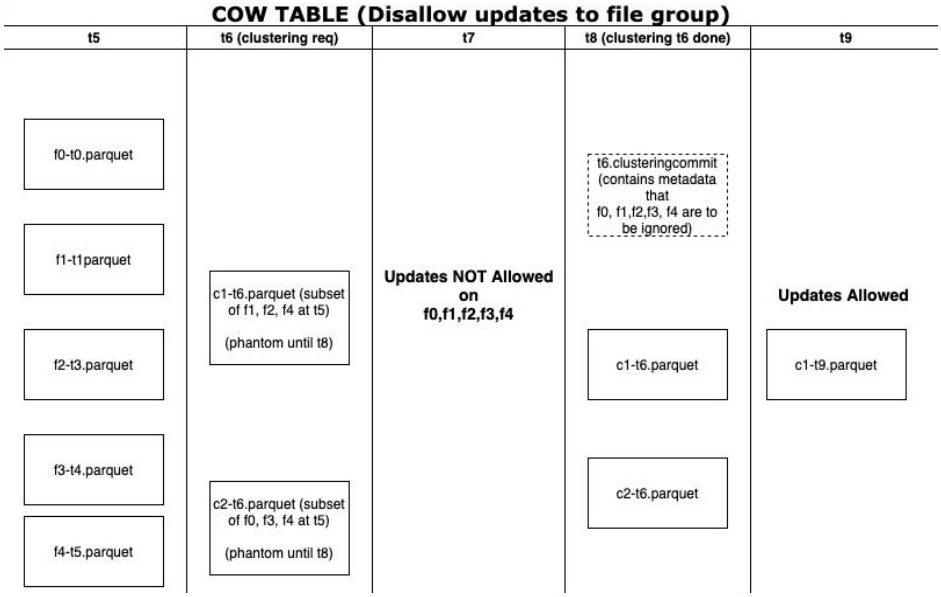
Architecture

Hudi provides different operations, such as **insert**, **upsert**, and **bulk_insert**, through its write client API to write data to a Hudi table. To weight between file size and speed of importing data into the data lake, Hudi provides **hoodie.parquet.small.file.limit** to configure the minimum file size. You can set it to **0** to force new data to be written to new file groups, or to a higher value to ensure that new data is "padded" to existing small file groups until it reaches the specified size, but this increases ingestion latency.

To support fast ingestion without affecting query performance, the clustering service is introduced to rewrite data to optimize the layout of Hudi data lake files.

The clustering service can run asynchronously or synchronously. It adds a new operation type called **REPLACE**, which will mark the clustering operation in the Hudi metadata timeline.

Clustering service is based on the MVCC design of Hudi to allow new data to be inserted. Clustering operations run in the background to reformat data layout, ensuring snapshot isolation between concurrent readers and writers.



Clustering is divided into two parts:

- Scheduling clustering: Create a clustering plan using a pluggable clustering strategy.
  a. Identify files that are eligible for clustering: Depending on the selected clustering strategy, the scheduling logic will identify the files eligible for clustering.
  b. Group files that are eligible for clustering based on specific criteria. The data size of each group must be a multiple of **targetFileSize**. Grouping is a part of the strategy defined in the plan. Additionally, there is an option to control group size to improve parallelism and avoid shuffling large volumes of data.
  c. Save the clustering plan to the timeline in Avro metadata format.
- Execute clustering: Process the plan using an execution strategy to create new files and replace old files.
  a. Read the clustering plan and obtain **clusteringGroups** that marks the file groups to be clustered.
  b. Instantiate appropriate strategy class for each group using **strategyParams** (for example, **sortColumns**) and apply the strategy to rewrite data.
  c. Create a **REPLACE** commit and update the metadata in HoodieReplaceCommitMetadata.

How to Execute Clustering

1. Executing clustering synchronously

   Add the following configuration parameters when the data write operation is performed:

   option("hoodie.clustering.inline", "true").

   option("hoodie.clustering.inline.max.commits", "4").

   option("hoodie.clustering.plan.strategy.target.file.max.bytes", "1073741824").

option("hoodie.clustering.plan.strategy.small.file.limit", "629145600").

option("hoodie.clustering.plan.strategy.sort.columns", "column1,column2").

2. Executing clustering asynchronously

   MRS 3.2.0 or later:

   Run the following spark-sql command to perform clustering. For details, see **CLUSTERING**.

   MRS 3.1.2:

   spark-submit --master yarn --class org.apache.hudi.utilities.HoodieClusteringJob /opt/client/Hudi/hudi/lib/hudi-utilities*.jar --schedule --base-path <table_path> --table-name <table_name> --props /tmp/clusteringjob.properties --spark-memory 1g

   spark-submit --master yarn --driver-memory 16G --executor-memory 12G --executor-cores 4 --num-executors 4 --class org.apache.hudi.utilities.HoodieClusteringJob /opt/client/Hudi/hudi/lib/hudi-utilities*.jar --base-path <table_path> --instant-time 20210605112954 --table-name <table_name> --props /tmp/clusteringjob.properties --spark-memory 12g

3. Specifying the ordering mode and sequence of clustering

   Currently, clustering supports three sorting modes: Linear, Z-Order, and Hilbert, which can be configured in option or set mode.

   - Linear ordering: a common but default ordering mode, which applies to ordering one field or multiple low-level fields.
   - Z-order or Hilbert: multi-dimensional ordering, which is available when you set **hoodie.layout.optimize.strategy** to **z-order** or **hilbert**.

     These two ordering modes apply to sorting 2 to 4 fields, for example, multiple fields involved in a query condition.

     Hilbert has a better multi-dimensional ordering effect than Z-order but lower ordering efficiency.

For details, see **Typical Hudi Configuration Parameters**.

CAUTION

1. The sorting column of clustering cannot be null. This is restricted by Spark RDD.
2. If the value of **target.file.max.bytes** is large, increase the value of **--spark-memory** to execute clustering. Otherwise, the executor memory overflow occurs.
3. Currently, the clean mechanism cannot be used to delete junk files generated after the clustering fails.
4. After the clustering, sizes of new files may be different, causing data skew.
5. Clustering and upsert operations cannot be performed at the same time.
6. If the clustering is in the **inflight** state, the files in the file group do not support the **update** operation.
7. If there is an unfinished clustering plan, an error will be reported when the compaction scheduling plan is generated upon subsequent writing. You must execute the clustering plan in a timely manner.

13.7.2 Hudi Cleaning

Cleaning is used to delete data of versions that are no longer required.

Hudi uses the cleaner working in the background to continuously delete unnecessary data of old versions. You can configure **hoodie.cleaner.policy** and **hoodie.cleaner.commits.retained** to use different cleaning policies and determine the number of saved commits.

You can use either of the following methods to perform cleaning:

- Synchronous cleaning is controlled by the **hoodie.clean.automatic** parameter, which is automatically enabled by default.

  Disable synchronous cleaning:

  When a data source is written, you can use **.option("hoodie.clean.automatic", "false")** to disable automatic cleaning.

  When spark-sql is written, you can use **set hoodie.clean.automatic=false;** to disable automatic cleaning.
- You can use spark-sql to perform asynchronous cleaning. For details, see **CLEAN**.

For more cleaning parameters, see **Compaction and Cleaning Configurations**.

13.7.3 Hudi Compaction

A compaction merges base and log files of MOR tables.

For MOR tables, data is stored in columnar Parquet files and row-based Avro files, updates are recorded in incremental files, and then a synchronous or asynchronous compaction is performed to generate new versions of columnar files. MOR tables can reduce data ingestion latency, so an asynchronous compaction that does not block ingestion is useful.

- An asynchronous compaction is performed in the following two steps:
  a. Scheduling a compaction: A compaction is completed by the job of importing data into the data lake. In this step, Hudi scans partitions and selects the file slices to be compacted. A compaction plan is finally written to the Hudi timeline.
  b. Executing a compaction: A separate process or thread reads the compaction plan and performs the compaction of file slices.
- Compaction can be synchronous or asynchronous.
  - The synchronization mode is controlled by the **hoodie.compact.inline** parameter. The default value is **true**, indicating that the compaction scheduling plan is automatically generated and compaction is executed.
    - Disable synchronous compaction.

      When a data source is written, run the **.option("hoodie.compact.inline", "false")** command to disable automatic compaction.

      When you use spark-sql to write data, run the **set hoodie.compact.inline=false;** command to disable automatic compaction.

- Only compaction scheduling is generated synchronously, but compaction is not executed.
  - · A data source can be written by configuring the following option parameters:

    option("hoodie.compact.inline", "true").

    option("hoodie.schedule.compact.only.inline", "true").

    option("hoodie.run.compact.only.inline", "false").
  - When you use·spark-sql to write data, configure the following **set** parameters:

    set hoodie.compact.inline=true;

    set hoodie.schedule.compact.only.inline=true;

    set hoodie.run.compact.only.inline=false;

– The asynchronous mode is implemented by spark-sql.

To execute only the compaction scheduling plan that has been generated during asynchronous compaction without creating a new scheduling plan, run the following commands to configure set parameters:

set hoodie.compact.inline=true;

set hoodie.schedule.compact.only.inline=false;

set hoodie.run.compact.only.inline=true;

For more compaction parameters, see **Compaction and Cleaning Configurations**.

NOTE

To ensure the maximum efficiency of data import into the lake, you are advised to generate compaction scheduling plans synchronously and execute compaction scheduling plans asynchronously.

13.7.4 Hudi Savepoint

Savepoints are used to save and restore custom version data.

Savepoints provided by Hudi can save different commits so that the cleaner program does not delete them. You can use rollback to restore them later.

Use spark-sql to manage savepoints.

Example:

- Creating a savepoint
  ```
  call create_savepoints('hudi_test1', '20220908155421949');
  ```
- Viewing all existing savepoints
  ```
  call show_savepoints(table =>'hudi_test1');
  ```
- Rolling back a savepoint
  ```
  call rollback_savepoints('hudi_test1', '20220908155421949');
  ```

NOTE

The **savepoint rollback** command is the same as the **commit rollback** commands. Both must be rolled back from the latest instant one by one.

MRS 3.1.2: MOR tables do not support savepoints.

13.7.5 Historical Hudi Data Deletion

NOTE

This topic is available for MRS 3.3.0-LTS and later versions only.

Scenario

Delete old data from Hudi tables to reduce space occupation and save storage costs.

Running delete/drop partition

The **delete/drop partition** command can be used to delete historical data. For details, see **Hudi SQL Syntax Reference**.

Advantages: The operation is simple and COW and MOR tables are supported.

Disadvantages: The concurrency is low. When Hudi tables are in the real-time write state, concurrent execution of the **delete/drop partition** command may cause the real-time data import job to fail.

Running call clean_data

- Function

  The **call clean_data** is used to delete historical data from MOR tables.

  Advantages: The deletion operation can be executed concurrently with the data import task, which does not affect the real-time import of data.

  Disadvantages: Only MOR tables are supported, and lazy deletion depends on compaction.

- Syntax

  call clean_data(table => 'table_name', sql => 'delete statement')

- Parameter description

  Table 13-4 Parameter description

Parameter	Description
table_name	Name of the table whose data is to be deleted. The value can be in the **database.tablename** format.
delete statement	SQL statement of the select type, which is used to find the data to be deleted.

- Example

  Delete all data whose **primaryKey** is smaller than 100 from the **mytable** table:

  ```
  call clean_data(table => 'mytable', sql=>'select *  from mytable where primaryKey < 100')
  ```

  Clear the residual files of the **clean_data** command. If **cleanData** fails, temporary files are generated. The following command can be used to clear these temporary files:


```
call clean_data(table => 'mytable', sql=>'delete cleanData')
```

- Response

  You can view query results on the client.

13.7.6 Hudi Payload

NOTE

This topic is available for MRS 3.3.0 or later only.

Introduction to Payload

Payload is the key for Hudi to implement incremental data update and deletion. It helps Hudi efficiently manage data changes in the data lake. The format of Hudi Payload is based on Apache Avro. It uses the Avro schema to define the data structure and type. Payloads can be serialized and deserialized so that data can be read and written in Hudi. In a word, Hudi Payload is an important part of Hudi. It provides a reliable, efficient, and scalable way to manage data changes in a large-scale data lake.

Typical Payload

- DefaultHoodieRecordPayload

  By default, DefaultHoodieRecordPayload is used in the Hudi. The payload compares the value of the **preCombineField** field in the incremental data with that in the inventory data to determine whether the inventory data with the same primary key can be updated by the incremental data with the same primary key. If the value of the **preCombineField** field in the incremental data with the same primary key is greater than that in the inventory data with the same primary key, the incremental data with the same primary key will be updated.

- OverwriteWithLatestAvroPayload

  The Payload ensures that the incremental data always overwrites the inventory data with the same primary key.

- PartialUpdateAvroPayload

  This payload inherits **OverwriteNonDefaultsWithLatestAvroPayload**, which ensures that null values in incremental data do not overwrite inventory data in any scenario.

Using Payload

- Specify a Payload during Spark table creation.

```
create table hudi_test(id int, comb int, price string, name string, par string) using hudi options(
primaryKey = "id",
preCombineField = "comb",
payloadClass="org.apache.hudi.common.model.OverwriteWithLatestAvroPayload") partitioned by
(par);
```

- Specify a Payload when data is written in Datasource mode.

```
data.write.format("hudi").
option("hoodie.datasource.write.table.type", COW_TABLE_TYPE_OPT_VAL).
option("hoodie.datasource.write.precombine.field", "comb").
option("hoodie.datasource.write.recordkey.field", "id").
option("hoodie.datasource.write.partitionpath.field", "par").
option("hoodie.datasource.write.payload.class",
```



```
"org.apache.hudi.common.model.DefaultHoodieRecordPayload").
option("hoodie.datasource.write.keygenerator.class", "org.apache.hudi.keygen.SimpleKeyGenerator").
option("hoodie.datasource.write.operation", "upsert").
option("hoodie.datasource.hive_sync.enable", "true").
option("hoodie.datasource.hive_sync.partition_fields", "par").
option("hoodie.datasource.hive_sync.partition_extractor_class",
"org.apache.hudi.hive.MultiPartKeysValueExtractor").
option("hoodie.datasource.hive_sync.table", "hudi_test").
option("hoodie.datasource.hive_sync.use_jdbc", "false").
option("hoodie.upsert.shuffle.parallelism", 4).
option("hoodie.datasource.write.hive_style_partitioning", "true").
option("hoodie.table.name", "hudi_test").mode(Append).save(s"/tmp/hudi_test")
```

13.8 Hudi SQL Syntax Reference

13.8.1 Restrictions on Using Hudi SQL

Hudi supports the DDL/DML syntax of using Spark SQL, making it easier for all users (non-engineers and analysts) to access and operate Hudi.

Constraints

- You can use Spark SQL to operate Hudi on the Hudi client.
- You can use Spark SQL to operate Hudi in JDBCServer of Spark2x.
- The Spark2x client does not support Spark SQL operations. The Spark3.1.1 or later client support Spark SQL operations.
- You cannot write data to Hudi tables or modify the Hudi table structure in Hive and Hetu engines. Only read operations are supported.
- The default value of **KeyGenerator** in SQL is **org.apache.hudi.keygen.ComplexKeyGenerator**. Therefore, you need to set the **KeyGenerator** value to that of SQL when data is written in DataSource mode.

13.8.2 Hudi DDL Syntax

13.8.2.1 CREATE TABLE

Function

This command is used to create a Hudi table by specifying the list of fields along with the table options.

Syntax

CREATE TABLE *[IF NOT EXISTS] [database_name.]table_name*

[(columnTypeList)]

USING hudi

[COMMENT table_comment]

[LOCATION location_path]

[OPTIONS (options_list)]

Parameter Description

Table 13-5 Parameters

Parameter	Description
database_name	Database name that contains letters, digits, and underscores (_).
table_name	Database table name that contains letters, digits, and underscores (_).
columnTypeList	List of comma-separated columns with data types. The column name contains letters, digits, and underscores (_).
using	Uses **hudi** to define and create a Hudi table.
table_comment	Description of the table.
location_path	HDFS path. If this parameter is set, the Hudi table will be created as an external table.
options_list	List of Hudi table options.

Table 13-6 Table options

Parameter	Description
primaryKey	Mandatory. Primary key name. Separate multiple primary key names with commas (,).
type	Type of the table. **"cow"** indicates a copy-on-write (COW) table, and **"mor"** indicates a merge-on-read (MOR) table. If this parameter is not specified, the default value is **"cow"**.
preCombineField	The **Pre-Combine** field in the table. This field is mandatory.
payloadClass	Logic that uses **preCombineField** for data filtering. **DefaultHoodieRecordPayload** is used by default. In addition, multiple preset payloads are provided, such as **OverwriteNonDefaultsWithLatestAvroPayload**, **OverwriteWithLatestAvroPayload**, and **EmptyHoodieRecordPayload**.
useCache	Whether to cache table relationships in Spark. This parameter does not need to be configured. This parameter is set to **false** by default to support the incremental view query of the COW table in Spark SQL.

Examples

- **Create a non-partitioned table.**

```
create table if not exists hudi_table0 (
id int,
name string,
price double
) using hudi
options (
type = 'cow',
primaryKey = 'id',
preCombineField = 'price'
);
```

- **Create a partitioned table.**

```
create table if not exists hudi_table_p0 (
id bigint,
name string,
ts bigint,
dt string,
hh string
) using hudi
options (
type = 'cow',
primaryKey = 'id',
preCombineField = 'ts'
)
partitioned by (dt, hh);
```

- **Create a table in a specified path.**

```
create table if not exists h3(
id bigint,
name string,
price double
) using hudi

options (
primaryKey = 'id',
preCombineField = 'price'
)
location '/path/to/hudi/h3';
```

- **Create a table and specify table attributes.**

```
create table if not exists h3(
id bigint,
name string,
price double
) using hudi
options (
primaryKey = 'id',
type = 'mor',
hoodie.cleaner.fileversions.retained = '20',
hoodie.keep.max.commits = '20'
);
```

Precautions

- Currently, Hudi does not support the CHAR, VARCHAR, TINYINT, and SMALLINT data types. You are advised to use the string or INT data type.
- Currently, only the following types of data support the configuration of default values: **int**, **bigint**, **float**, **double**, **decimal**, **string**, **date**, **timestamp**, **boolean**, and **binary**.
- You must specify **primaryKey** and **preCombineField** for Hudi tables.
- When you create a table in a specified path and there already are Hudi tables in the path, you do not need to specify columns during table creation.

System Response

The table is successfully created, and the success message is logged in the system.

13.8.2.2 CREATE TABLE AS SELECT

Function

This command is used to create a Hudi table by specifying the list of fields along with the table options.

Syntax

CREATE TABLE *[IF NOT EXISTS] [database_name.]table_name*

USING hudi

[COMMENT table_comment]

[LOCATION location_path]

[OPTIONS (options_list)]

[AS query_statement]

Parameter Description

Table 13-7 Parameters

Parameter	Description
database_name	Database name that contains letters, digits, and underscores (_).
table_name	Database table name that contains letters, digits, and underscores (_).
using	Uses **hudi** to define and create a Hudi table.
table_comment	Description of the table.
location_path	HDFS path. If this parameter is set, the Hudi table will be created as an external table.
options_list	List of Hudi table options.
query_statement	SELECT statement

Examples

- Create a partitioned table.

```
create table h2 using hudi
options (type = 'cow', primaryKey = 'id')
partitioned by (dt)
as
select 1 as id, 'a1' as name, 10 as price, 1000 as dt;
```


- Create a non-partitioned table.

```
create table h3 using hudi
as
select 1 as id, 'a1' as name, 10 as price;

Load data from a parquet table to the Hudi table.
# Create a parquet table.
create table parquet_mngd using parquet options(path='hdfs:///tmp/parquet_dataset/*.parquet');

# Create a Hudi table by Creating a Table from Query Results (CTAS).
create table hudi_tbl using hudi location 'hdfs:///tmp/hudi/hudi_tbl/' options (
type = 'cow',
primaryKey = 'id',
preCombineField = 'ts'
)
partitioned by (datestr) as select * from parquet_mngd;
```

Precautions

For better data loading performance, CTAS uses **bulk insert** to write data.

System Response

The table is successfully created, and the success message is logged in the system.

13.8.2.3 DROP TABLE

Function

This command is used to delete an existing table.

Syntax

DROP TABLE *[IF EXISTS] [db_name.]table_name;*

Parameter Description

Table 13-8 Parameters

Parameter	Description
db_name	Database name. If this parameter is not specified, the current database is selected.
table_name	Name of the table to be deleted.

Precautions

In this command, **IF EXISTS** and **db_name** are optional.

Examples

DROP TABLE IF EXISTS hudidb.h1;

System Response

The table will be deleted.

13.8.2.4 SHOW TABLE

Function

This command is used to display all tables in current database or all tables in a specific database.

Syntax

SHOW TABLES *[IN db_name];*

Parameter Description

Table 13-9 Parameters

Parameter	Description
IN db_name	Name of the specific database whose tables need to be all displayed.

Precautions

IN db_Name is optional. It is required only when you need to display all tables of a specific database.

Examples

```
SHOW TABLES IN hudidb;
```

System Response

All tables are listed.

13.8.2.5 ALTER RENAME TABLE

NOTE

This section applies only to MRS 3.2.0-LTS to MRS 3.3.1-LTS.

Function

This command is used to rename an existing table.

Syntax

ALTER TABLE *oldTableName* **RENAME TO** *newTableName*

Parameter Description

Table 13-10 Parameters

Parameter	Description
oldTableName	Current name of the table
new_table_name	New name of the table

Examples

```
alter table h0 rename to h0_1;
```

System Response

The table name is changed. You can run the **SHOW TABLES** command to display the new table name.

13.8.2.6 ALTER ADD COLUMNS

NOTE

This section applies only to MRS 3.2.0-LTS to MRS 3.3.1-LTS.

Function

This command is used to add columns to an existing table.

Syntax

ALTER TABLE *tableIdentifier* **ADD COLUMNS***(colAndType (,colAndType)*)*

Parameter Description

Table 13-11 Parameters

Parameter	Description
tableIdentifier	Table name.
colAndType	Name of a comma-separated column with data types. A column name consists of letters, digits, and underscores (_).

Examples

```
alter table h0_1 add columns(ext0 string);
```


System Response

The columns are added to the table. You can run the **DESCRIBE** command to display the columns.

13.8.2.7 ALTER COLUMN

NOTE

This section applies only to MRS 3.3.0-LTS and MRS 3.3.1-LTS.

Function

This command is used to change the default values of a column.

Syntax

ALTER TABLE *tableIdentifier* **ALTER COLUMN** *colName* **SET DEFAULT** *defaultValue*

Parameters

Table 13-12 ADD COLUMNS parameters

Parameter	Description
tableIdentifier	ClickHouse table name
colName	Column Name
defaultValue	Default value of a column

Example

```
alter table h0_1 alter column extl set default 'new_default_value';
```

Response

You can view query results on the client.

13.8.2.8 TRUNCATE TABLE

Function

This command is used to clear all data in a specific table.

Syntax

TRUNCATE TABLE *tableIdentifier*

Parameter Description

Table 13-13 Parameters

Parameter	Description
tableIdentifier	Table name.

Examples

```
truncate table h0_1;
```

System Response

Data in the table is cleared. You can run the **QUERY** statement to check whether data in the table has been deleted.

13.8.3 Hudi DML Syntax

13.8.3.1 INSERT INTO

Function

This command is used to insert the output of the **SELECT** statement to a Hudi table.

Syntax

INSERT INTO *tableIndentifier select query;*

Parameter Description

Table 13-14 Parameters

Parameter	Description
tableIndentifier	Name of the Hudi table.
select query	SELECT statement.

Precautions

- Insert mode: Hudi supports three insert modes for tables with primary keys. You can set **hoodie.sql.insert.mode** to specify the insert mode. The default value is **upsert**.
  - In strict mode, the INSERT statement retains the primary key constraint of COW tables and is not allowed to insert duplicate records. If a record already exists during data insertion, HoodieDuplicateKeyException is

thrown for COW tables. For MOR tables, the behavior in this mode is the same as that in upsert mode.

- In non-strict mode, records are inserted to primary key tables.
- In upsert mode, duplicate values in the primary key table are updated.

- When executing a SQL statement, you can set **hoodie.sql.bulk.insert.enable** to **true** and **hoodie.sql.insert.mode** to **non-strict** to enable the bulk insert statement as the write mode of the insert statement.

  You can also set **hoodie.datasource.write.operation** to control the write mode of the insert statement, including **bulk_insert**, **insert**, and **upsert**. If you use this setting method, you must run **reset hoodie.datasource.write.operation;** to reset the Hudi write mode after executing the SQL statement. Otherwise, this parameter may affect the execution of other SQL statements.

Examples

```
insert into h0 select 1, 'a1', 20;

-- insert static partition
insert into h_p0 partition(dt = '2021-01-02') select 1, 'a1';

-- insert dynamic partition
insert into h_p0 select 1, 'a1', dt;

-- insert dynamic partition
insert into h_p1 select 1 as id, 'a1', '2021-01-03' as dt, '19' as hh;

-- insert overwrite table
insert overwrite table h0 select 1, 'a1', 20;

-- insert overwrite table with static partition
insert overwrite h_p0 partition(dt = '2021-01-02') select 1, 'a1';

-- insert overwrite table with dynamic partition
insert overwrite table h_p1 select 2 as id, 'a2', '2021-01-03' as dt, '19' as hh;
```

System Response

You can view the result in driver logs.

13.8.3.2 MERGE INTO

Function

This command is used to query another table based on the join condition of a table or subquery. If **UPDATE** or **DELETE** is executed for the table matching the join condition, and **INSERT** is executed if the join condition is not met. This command completes the synchronization requiring only one full table scan, delivering higher efficiency than **INSERT** plus **UPDATE**.

Syntax

MERGE INTO *tableIdentifier AS target_alias*

USING *(sub_query | tableIdentifier) AS source_alias*

ON *<merge_condition>*

[WHEN MATCHED [AND <condition>] THEN <matched_action>]

[WHEN MATCHED [AND <condition>] THEN <matched_action>]

[WHEN NOT MATCHED [AND <condition>] THEN <not_matched_action>]

<merge_condition> =A equal bool condition

<matched_action> =

DELETE |

*UPDATE SET * |*

UPDATE SET column1 = expression1 [, column2 = expression2 ...]

<not_matched_action> =

*INSERT * |*

INSERT (column1 [, column2 ...]) VALUES (value1 [, value2 ...])

Parameter Description

Table 13-15 Parameters

Parameter	Description
tableIdentifier	Name of the Hudi table.
target_alias	Alias of the target table.
sub_query	Subquery.
source_alias	Alias of the source table or source expression.
merge_condition	Condition for associating the source table or expression with the target table.
condition	Filtering condition. This parameter is optional.
matched_action	DELETE or UPDATE operation to be performed when conditions are met.
not_matched_action	INSERT operation to be performed when conditions are not met.

Precautions

1. The merge-on condition supports only primary key columns currently.
2. Currently, only some fields in the COW table can be updated, and the updated values must contain the pre-merged columns. All fields in the MOR table must be provided in the Update statement.

Examples

- Update partial fields only.

```
create table h0(id int, comb int, name string, price int) using hudi options(primaryKey = 'id',
preCombineField = 'comb');
create table s0(id int, comb int, name string, price int) using hudi options(primaryKey = 'id',
preCombineField = 'comb');
insert into h0 values(1, 1, 1, 1);
insert into s0 values(1, 1, 1, 1);
insert into s0 values(2, 2, 2, 2);
// Method 1
merge into h0 using s0
on h0.id = s0.id
when matched then update set h0.id = s0.id, h0.comb = s0.comb, price = s0.price * 2;
// Method 2
merge into h0 using s0
on h0.id = s0.id
when matched then update set id = s0.id,
name = h0.name,
comb = s0.comb + h0.comb,
price = s0.price + h0.price;
```

- Update and insert default fields.

```
create table h0(id int, comb int, name string, price int, flag boolean) using hudi options(primaryKey =
'id', preCombineField = 'comb');
create table s0(id int, comb int, name string, price int, flag boolean) using hudi options(primaryKey =
'id', preCombineField = 'comb');
insert into h0 values(1, 1, 1, 1, false);
insert into s0 values(1, 2, 1, 1, true);
insert into s0 values(2, 2, 2, 2, false);

merge into h0 as target
using (
select id, comb, name, price, flag from s0
) source
on target.id = source.id
when matched then update set *
when not matched then insert *;
```

- Update and delete data based on multiple conditions.

```
create table h0(id int, comb int, name string, price int, flag boolean) using hudi options(primaryKey =
'id', preCombineField = 'comb');
create table s0(id int, comb int, name string, price int, flag boolean) using hudi options(primaryKey =
'id', preCombineField = 'comb');
insert into h0 values(1, 1, 1, 1, false);
insert into h0 values(2, 2, 1, 1, false);
insert into s0 values(1, 1, 1, 1, true);
insert into s0 values(2, 2, 2, 2, false);
insert into s0 values(3, 3, 3, 3, false);

merge into h0
using (
select id, comb, name, price, flag from s0
) source
on h0.id = source.id
when matched and flag = false then update set id = source.id, comb = h0.comb + source.comb, price =
source.price * 2
when matched and flag = true then delete
when not matched then insert *;
```

System Response

You can view the result in driver logs or on the client.

13.8.3.3 UPDATE

Function

This command is used to update the Hudi table based on the column expression and optional filtering conditions.

Syntax

UPDATE *tableIdentifier SET column = EXPRESSION(,column = EXPRESSION) [WHERE boolExpression]*

Parameter Description

Table 13-16 Parameters

Parameter	Description
tableIdentifier	Name of the Hudi table to be updated.
column	Target column to be updated.
EXPRESSION	Expression of the source table column to be updated in the target table.
boolExpression	Filtering condition expression.

Examples

```
update h0 set price = price + 20 where id = 1;
update h0 set price = price *2, name = 'a2' where id = 2;
```

System Response

You can view the result in driver logs or on the client.

13.8.3.4 DELETE

Function

This command is used to delete records from a Hudi table.

Syntax

DELETE from *tableIdentifier [WHERE boolExpression]*

Parameter Description

Table 13-17 Parameters

Parameter	Description
tableIdentifier	Name of the Hudi table to delete records.
boolExpression	Filtering conditions for deleting records.

Examples

- Example 1:
  ```
  delete from h0 where column1 = 'country';
  ```
- Example 2:
  ```
  delete from h0 where column1 IN ('country1', 'country2');
  ```
- Example 3:
  ```
  delete from h0 where column1 IN (select column11 from sourceTable2);
  ```
- Example 4:
  ```
  delete from h0 where column1 IN (select column11 from sourceTable2 where column1 = 'xxx');
  ```
- Example 5:
  ```
  delete from h0;
  ```

System Response

You can view the result in driver logs or on the client.

13.8.3.5 COMPACTION

Function

This command is used to convert row-based log files in MOR tables into column-based data files in parquet tables, accelerating record search.

Syntax

SCHEDULE COMPACTION on *tableIdentifier* |tablelocation;

SHOW COMPACTION on *tableIdentifier* |tablelocation;

RUN COMPACTION on *tableIdentifier* |tablelocation [at instant-time];

Parameter Description

Table 13-18 Parameters

Parameter	Description
tableIdentifier	Name of the Hudi table to convert log files.
tablelocation	Storage path of the Hudi table.

Parameter	Description
instant-time	Time to run the command. You can run the **show compaction** command to view the **instant-time** value.

Examples

```
schedule compaction  on h1;
show compaction on h1;
run compaction on h1 at 20210915170758;

schedule compaction  on '/tmp/hudi/h1';
run compaction on '/tmp/hudi/h1';
```

Precautions

You need to set **hoodie.payload.ordering.field** to the value of **preCombineField** when you use the Hudi CLI or API to trigger compaction for the Hudi table created by SQL.

System Response

You can view the result in driver logs or on the client.

13.8.3.6 SET/RESET

Function

This command is used to dynamically add, update, display, or reset Hudi parameters without restarting the driver.

Syntax

- Add or update a parameter value:

  SET *parameter_name=parameter_value*

  This command is used to add or update the value of **parameter_name**.
- Display a parameter value:

  SET *parameter_name*

  This command is used to display the value of **parameter_name**.
- Display session parameters:

  SET

  This command is used to display all supported session parameters.
- Display session parameters along with usage details:

  SET -v

  This command is used to display all supported session parameters and their usage details.
- Reset parameter values:

  RESET

This command is used to reset all session parameters.

Parameter Description

Table 13-19 Parameters

Parameter	Description
parameter_name	Name of the parameter to be dynamically added, updated, or displayed.
parameter_value	New value to be set for **parameter_name**.

Precautions

The following table lists the properties to be used in the **SET** or **RESET** commands.

Table 13-20 Properties

Property	Description
hoodie.insert.shuffle.parallelism	Degree of parallelism (DOP) of Spark shuffle for writing data in insert mode.
hoodie.upsert.shuffle.parallelism	DOP of Spark shuffle for writing data in upsert mode.
hoodie.delete.shuffle.parallelism	DOP of Spark shuffle for deleting data in delete mode.
hoodie.sql.insert.mode	Insert mode. The value can be strict, non-strict, or upsert.
hoodie.sql.bulk.insert.enable	Whether to enable bulk insert.
spark.sql.hive.convertMetastoreParquet	Converts the parquet table into a data source table for reading. If the provider of Hudi is Hive and Spark SQL or Spark Beeline is used to read data, set this parameter to **false**.

Examples

- Add or Update command:

```
set hoodie.insert.shuffle.parallelism = 100;
set hoodie.upsert.shuffle.parallelism = 100;
set hoodie.delete.shuffle.parallelism = 100;
```

- Reset command:

```
RESET
```

System Response

- You can view the success result in driver logs.

- You can view the failure result on the UI.

13.8.3.7 ARCHIVELOG

NOTE

This section applies only to MRS 3.2.0 or later.

Function

Archives instants on the Timeline based on configurations and deletes archived instants from the Timeline to reduce the operation pressure on the Timeline.

Syntax

RUN ARCHIVELOG ON tableIdentifier;

RUN ARCHIVELOG ON tablelocation;

Parameter Description

Table 13-21 Parameters

Parameter	Description
tableIdentifier	Name of the Hudi table
tablelocation	Storage path of the Hudi table

Example

```
run archivelog on h1;
run archivelog on "/tmp/hudi/h1";
```

Precautions

- Only instants that are not cleaned can be archived.
- No matter whether the compaction operation is performed, at least *x* (*x* indicates the value of **hoodie.compact.inline.max.delta.commits**) instants are retained and not archived to ensure that there are enough instants to trigger the compaction schedule.

System Response

You can view command execution results in the driver log or on the client.

13.8.3.8 CLEAN

NOTE

This section applies only to MRS 3.2.0 or later.

Function

Cleans instants on the Timeline based on configurations and deletes historical version files to reduce the data storage and read/write pressure of Hudi tables.

Syntax

RUN CLEAN ON tableIdentifier;

RUN CLEAN ON tablelocation;

Parameter Description

Table 13-22 Parameters

Parameter	Description
tableIdentifier	Name of the Hudi table
tablelocation	Storage path of the Hudi table

Example

```
run clean on h1;
run clean on "/tmp/hudi/h1";
```

Precautions

Only the table owner can perform the clean operation on a table.

To modify the default cleaning parameters, run **set** commands to configure the parameters such as the number of commits to be retained.

System Response

You can view command execution results in the driver log or on the client.

13.8.3.9 CLEANARCHIVE

NOTE

This section applies only to MRS 3.2.1-LTS to MRS 3.3.1-LTS.

Function

Deletes the archive files of Hudi tables to reduce data storage and read/write pressure of Hudi tables.

Syntax

set hoodie.archive.file.cleaner.policy = KEEP_ARCHIVED_FILES_BY_SIZE;

set hoodie.archive.file.cleaner.size.retained = 5368709120;

run cleanarchive on *tableIdentifier*|*tablelocation*;

set hoodie.archive.file.cleaner.policy = KEEP_ARCHIVED_FILES_BY_DAYS;

set hoodie.archive.file.cleaner.days.retained = 30;

run cleanarchive on *tableIdentifier*|*tablelocation*;

Parameter Description

Table 13-23 Parameters

Parameter	Description
tableIdentifier	Name of the Hudi table
tablelocation	Storage path of the Hudi table
hoodie.archive.file.cleaner.policy	Policy for clearing archived files: Currently, only the **KEEP_ARCHIVED_FILES_BY_SIZE** and **KEEP_ARCHIVED_FILES_BY_DAYS** policies are supported. The default policy is **KEEP_ARCHIVED_FILES_BY_DAYS**. • **KEEP_ARCHIVED_FILES_BY_SIZE**: used to configure the storage capacity that can be used by archived files. • **KEEP_ARCHIVED_FILES_BY_DAYS**: used to delete archived files beyond a specified time point.
hoodie.archive.file.cleaner.size.retained	When the deletion policy is **KEEP_ARCHIVED_FILES_BY_SIZE**, this parameter specifies the number of bytes of archived files to be retained. The default value is **5368709120** bytes (5 GB).
hoodie.archive.file.cleaner.days.retained	When the deletion policy is **KEEP_ARCHIVED_FILES_BY_DAYS**, this parameter specifies the number of days for storing archived files. The default value is **30** days.

Precautions

Archived files are not backed up and cannot be restored after being deleted.

System Response

You can view command execution results in the driver log or on the client.

13.8.3.10 Drop Partition

NOTE

This section applies only to MRS 3.50-LTS or later.

Function

This command is used to delete a single partition or multiple partitions.

Syntax

alter table $tableName drop partition *(Partitioning field= 'Match condition')*

Example

```
// Delete a partition.
alter table $tableName drop partition (dt='2021-10-01')
// Delete partitions.
alter table $tableName drop partition (dt='2021-10-01', dt='2021-10-02')
// Fuzzy match to delete multiple partitions.
alter table $tableName drop partition (dt='2021-10-*') // Delete the partition of October.
```

Precautions

You can use fuzzy match to delete multiple partitions. Only asterisks (*) are supported. Complex regular expressions are not supported.

Response

You can view command execution results in the driver log or on the client.

13.8.4 Hudi CALL COMMAND Syntax

13.8.4.1 CHANGE_TABLE

The Hudi CALL COMMAND syntax is available in MRS 3.2.0 and later versions.

Function

Modifies the type and index of a table. Key parameters such as the type and index of Hudi tables cannot be modified. Therefore, this command is actually used to rewrite Hudi tables.

Syntax

```
call change_table(table => '[table_name]', hoodie.index.type => '[index_type]',
hoodie.datasource.write.table.type => '[table_type]');
```


Parameter Description

Table 13-24 Parameters

Parameter	Description
table_name	Name of the table to be modified
table_type	Type of the table to be modified
index_type	Type of the index to be modified

Precautions

If the index type to be modified has other configuration parameters, the parameters must be transferred to the SQL statement in the **key =>'value'** format.

For example, to change the index type to bucket, run the following command:

```
call change_table(table => 'hudi_table1', hoodie.index.type => 'BUCKET', hoodie.bucket.index.num.buckets => '3');
```

Example

call change_table(table => 'hudi_table1', hoodie.index.type => 'SIMPLE', hoodie.datasource.write.table.type => 'MERGE_ON_READ');

System Response

After the execution is complete, you can run the **desc formatted table** command to view the table properties.

13.8.4.2 CLEAN_FILE

Function

Cleans invalid data files from the Hudi table directory.

Syntax

call clean_file(table => '[table_name]', mode=>'[op_type]', backup_path=>'[backup_path]', start_instant_time=>'[start_time]', end_instant_time=>'[end_time]');

Parameter Description

Table 13-25 Parameters

Parameter	Description
table_name	Mandatory. Name of the Hudi table from which invalid data files are to be deleted.
op_type	Optional. Command running mode. The default value is **dry_run**. Value options are **dry_run**, **repair**, **undo**, and **query**. **dry_run**: displays invalid data files to be cleaned. **repair**: displays and cleans invalid data files. **undo**: restores deleted data files. **query**: displays the backup directories that have been cleaned.
backup_path	Mandatory. Backup directory of the data files to be restored. This parameter is available only when the running mode is **undo**.
start_time	Optional. Start time for generating invalid data files. This parameter is available only when the running mode is **dry_run** or **repair**. The start time is not limited by default.
end_time	Optional. End time for generating invalid data files. This parameter is available only when the running mode is **dry_run** or **repair**. The end time is not limited by default.

Example

```
call clean_file(table => 'h1', mode=>'repair');
call clean_file(table => 'h1', mode=>'dry_run');
call clean_file(table => 'h1', mode=>'query');
call clean_file(table => 'h1', mode=>'undo', backup_path=>'/tmp/hudi/h1/.hoodie/.cleanbackup/
hoodie_repair_backup_20220222222222');
```

Precautions

The command cleans only invalid Parquet files.

System Response

You can view command execution results in the driver log or on the client.

13.8.4.3 SHOW_TIME_LINE

Function

Displays the effective or archived Hudi timelines and details of a specified instant time.

Syntax

- Viewing the list of effective timelines of a table:
  call show_active_instant_list(table => '[table_name]');
- Viewing the list of effective timelines after a timestamp in a table:
  call show_active_instant_list(table => '[table_name]', instant => '[instant]');
- Viewing information about an instant that takes effect in a table:
  call show_active_instant_detail(table => '[table_name]', instant => '[instant]');
- Viewing the list of archived instant timelines in a table:
  call show_archived_instant_list(table => '[table_name]');
- Viewing the list of archived instant timelines after a timestamp in a table:
  call show_archived_instant_list(table => '[table_name]', instant => '[instant]');
- Viewing information about archived instants in a table:
  call show_archived_instant_detail(table => '[table_name], instant => '[instant]');

Parameter Description

Table 13-26 Parameters

Parameter	Description
table_name	Name of the table to be queried. The value can be in the *database*.*tablename* format.
instant	Instant timestamp to be queried

Example

```
call show_active_instant_detail(table => 'hudi_table1', instant => '20220913144936897'");
```

System Response

You can view query results on the client.

13.8.4.4 SHOW_HOODIE_PROPERTIES

Function

Displays the configuration in the **hoodie.properties** file of a specified Hudi table.

Syntax

call show_hoodie_properties(table => '[table_name]');

Parameter Description

Table 13-27 Parameters

Parameter	Description
table_name	Name of the table to be queried. The value can be in the *database*.*tablename* format.

Example

```
call show_hoodie_properties(table => "hudi_table5");
```

System Response

You can view query results on the client.

13.8.4.5 SAVE_POINT

Function

Manages savepoints of Hudi tables.

Syntax

- Creating a savepoint:

  call create_savepoints('[table_name]', '[commit_Time]', '[user]', '[comments]');
- Viewing all existing savepoints

  call show_savepoints(table =>'[table_name]');
- Rolling back a savepoint:

  call rollback_savepoints('[table_name]', '[commit_Time]');

Parameter Description

Table 13-28 Parameters

Parameter	Description	Mandatory
table_name	Name of the table to be queried. The value can be in the *database.tablename* format.	Yes
commit_Time	Specified creation or rollback timestamp	Yes
user	User who creates a savepoint	No
comments	Description of the savepoint	No

Example

```
call create_savepoints('hudi_test1', '20220908155421949');
call show_savepoints(table =>'hudi_test1');
call rollback_savepoints('hudi_test1', '20220908155421949');
```

Precautions

- MOR tables do not support savepoints.
- The commit-related files before the latest savepoint are not cleaned.
- If there are multiple savepoints, perform the rollback from the latest savepoint. The logic is as follows: roll back the latest savepoint; delete the savepoint; and roll back the next savepoint.

System Response

You can view query results on the client.

13.8.4.6 ROLL_BACK

Function

Rolls back a specified commit.

Syntax

call rollback_to_instant(table => '[table_name]', instant_time => '[instant]');

Parameter Description

Table 13-29 Parameters

Parameter	Description
table_name	Mandatory. Name of the Hudi table to be rolled back.
instant	Mandatory. Commit instant timestamp of the Hudi table to be rolled back.

Example

```
call rollback_to_instant(table => 'h1', instant_time=>'20220915113127525');
```

Precautions

Only the latest commit timestamps can be rolled back in sequence.

System Response

You can view command execution results in the driver log or on the client.

13.8.4.7 CLUSTERING

NOTE

This topic is available for MRS 3.2.0 or later only.

Function

Performs the clustering operation on Hudi tables. For details, see **Hudi Clustering**.

Syntax

- Performing clustering:

  call run_clustering(table=>'[table]', path=>'[path]', predicate=>'[predicate]', order=>'[order]');
- Viewing the clustering plan:

  call show_clustering(table=>'[table]', path=>'[path]', limit=>'[limit]');

Parameter Description

Table 13-30 Parameters

Parameter	Description	Mandatory
table	Name of the table to be queried. The value can be in the *database.tablename* format.	No
path	Path of the table to be queried	No
predicate	Predicate to be defined	No
order	Sorting field for clustering	No
limit	Number of query results to display	No

Example

```
call show_clustering(table => 'hudi_table1');

call run_clustering(table => 'hudi_table1', predicate => '(ts >= 1006L and ts < 1008L) or ts >= 1009L', order => 'ts');

call run_clustering(path => '/user/hive/warehouse/hudi_test2', predicate => "dt = '2021-08-28'", order => 'id');
```

Precautions

- Either **table** or **path** must exist. Otherwise, the Hudi table to be clustered cannot be determined.
- To cluster a specified partition, refer to the format **predicate => "dt = '2021-08-28'"**.

System Response

You can view query results on the client.

13.8.4.8 CLEANING

NOTE

This topic is available for MRS 3.3.0 or later only.

Function

Cleans Hudi tables. For details, see **Hudi Cleaning**.

Syntax

call run_clean(table=>'[table]', clean_policy=>'[clean_policy]', retain_commits=>'[retain_commits]', hours_retained=> '[hours_retained]', file_versions_retained=> '[file_versions_retained]');

Parameters

Table 13-31 Parameter description

Parameter	Description	Mandatory
table	Name of the table to be queried. The value can be in the *database*.*tablename* format.	Yes
clean_policy	Policy for deleting data files of an earlier version. The default value is **KEEP_LATEST_COMMITS**.	No
retain_commits	This parameter is available only when **clean_policy** is set to **KEEP_LATEST_COMMITS**.	No
hours_retained	This parameter is available only when **clean_policy** is set to **KEEP_LATEST_BY_HOURS**.	No
file_version_retained	This parameter is available only when **clean_policy** is set to **KEEP_LATEST_FILE_VERSIONS**.	No

Example

```
call run_clean(table => 'hudi_table1');

call run_clean(table => 'hudi_table1', retain_commits => 2);

call run_clean(table => 'hudi_table1', clean_policy => 'KEEP_LATEST_FILE_VERSIONS', file_version_retained => 1);
```

Constraints

The cleaning operation cleans data files of an earlier version in partitions only when trigger conditions are met. If trigger conditions are not met, this operation does not clean the data files even if the command is successfully executed.

Response

You can view query results on the client.

13.8.4.9 COMPACTION

NOTE

This topic is available for MRS 3.3.0 or later only.

Function

Compacts Hudi tables. For details, see **Hudi Compaction**.

Syntax

call run_compaction(op => '[op]', table=>'[table]', path=>'[path]', timestamp=>'[timestamp]');

Parameters

Table 13-32 Parameter description

Parameter	Description	Mandatory
op	Set this parameter to **schedule** to generate a compaction plan or to **run** to execute a generated compaction plan.	Yes
table	Name of the table to be queried. The value can be in the *database.tablename* format.	No
path	Path of the table to be queried	No
timestamp	When **op** is set to **run**, you can specify **timestamp** to execute the compaction plan corresponding to the timestamp and the compaction plan that is not executed before the timestamp.	No

Example

```
call run_compaction(table => 'hudi_table1', op => 'schedule');
call run_compaction(table => 'hudi_table1', op => 'run');
call run_compaction(table => 'hudi_table1', op => 'run', timestamp => 'xxx');
call run_compaction(path => '/user/hive/warehouse/hudi_table1', op => 'run', timestamp => 'xxx');
```

Precautions

Only MOR tables can be compacted.

Response

You can view query results on the client.

13.8.4.10 SHOW_COMMIT_FILES

NOTE

This topic is available for MRS 3.3.0 or later only.

Function

Checks whether multiple files are updated in or inserted to a specified instant.

Syntax

call show_commit_files(table=>'[table]', instant_time=>'[instant_time]', limit=>'[limit]');

Parameters

Table 13-33 Parameter description

Parameter	Description	Mandatory
table	Name of the table to be queried. The value can be in the *database.tablename* format.	Yes
instant_time	Timestamp corresponding to a commit operation	Yes
limit	Number of returned items	No

Example

```
call show_commit_files(table=>'hudi_mor', instant_time=>'20230216144548249');
call show_commit_files(table=>'hudi_mor', instant_time=>'20230216144548249', limit=>'1');
```

Returned Result

Parameter	Description
action	Action type of the commit operation corresponding to **instant_time**, such as **compaction**, **deltacommit**, and **clean**
partition_path	Partition where the file updated in or inserted to a specified instant is located
file_id	ID of the file updated in or inserted to the specified instant

Parameter	Description
previous_commit	Timestamp in the file name updated or inserted by the specified instant.
total_records_updated	Number of updated records in the file
total_records_written	Number of records inserted into the file
total_bytes_written	Number of bytes of data added to the file
total_errors	Total number of errors reported during the update in or insertion to the specified instant
file_size	File size, in bytes

Response

You can view query results on the client.

13.8.4.11 SHOW_FS_PATH_DETAIL

NOTE

This topic is available for MRS 3.3.0 or later only.

Function

Checks statistics about a specified FS path.

Syntax

call show_fs_path_detail(path=>'[path]', is_sub=>'[is_sub]', sort=>'[sort]');

Parameters

Table 13-34 Parameter description

Parameter	Description	Mandatory
path	Path of the FS to be queried	Yes
is_sub	The default value is **false**, indicating that statistics about a specified directory is collected. The value **true** indicates that statistics about subdirectories in a specified directory is collected.	No

Parameter	Description	Mandatory
sort	The default value is **true**, indicating that the results are sorted based on **storage_size**. The value **false** indicates that the results are sorted based on the number of files.	No

Example

```
call show_fs_path_detail(path=>'/user/hive/warehouse/hudi_mor/dt=2021-08-28', is_sub=>false, sort=>true);
```

Returned Result

Parameter	Description
path_num	Number of subdirectories in a specified directory
file_num	Number of files in a specified directory
storage_size	Size of the directory, in bytes
storage_size(unit)	Size of the directory, in KB
storage_path	Complete FS absolute path of the specified directory
space_consumed	Actual space occupied by the returned files/directories in the cluster, that is, the replication factor set for the cluster is considered.
quota	Name quota, which is a mandatory restriction on the number of files and directory names in the current directory tree
space_quota	Space quota, which is a mandatory restriction on the number of bytes used by files in the current directory tree

Response

You can view query results on the client.

13.8.4.12 SHOW_LOG_FILE

NOTE

This topic is available for MRS 3.3.0 or later only.

Function

This command is used to view the meta and record information in log files.

Syntax

- Checking meta information:

  call show_logfile_metadata(table => '[table]', log_file_path_pattern => '[log_file_path_pattern]', limit => '[limit]')

- Checking record information:

  call show_logfile_records(table => '[table]', log_file_path_pattern => '[log_file_path_pattern]', merge => '[merge]', limit => '[limit]')

Parameters

Table 13-35 Parameter description

Parameter	Description	Mandatory
table	Name of the table to be queried. The value can be in the *database*.*tablename* format.	Yes
log_file_path_pattern	Path of log files. Regular expression matching is supported.	No
merge	When **show_logfile_records** is executed, this parameter is used to specify whether to combine records in multiple log files and return them together.	No
limit	Number of returned items	No

Example

```
call show_logfile_metadata(table => 'hudi_mor', log_file_path_pattern => 'http://hacluster/user/hive/warehouse/hudi_mor/dt=2021-08-28/.*?log.*?');
call show_logfile_records(table => 'hudi_mor', log_file_path_pattern => 'http://hacluster/user/hive/warehouse/hudi_mor/dt=2021-08-28/.*?log.*?', merge => false, limit => 1);
```

Constraints

- This command is used only for MOR tables.

Response

You can view query results on the client.

13.8.4.13 SHOW_INVALID_PARQUET

NOTE

This topic is available for MRS 3.3.0 or later only.

Function

Checks the damaged parquet file in the execution path.

Syntax

call show_invalid_parquet(path => 'path')

Parameters

Table 13-36 Parameter description

Parameter	Description	Mandatory
path	Path of the FS to be queried	Yes

Example

```
call show_invalid_parquet(path => '/user/hive/warehouse/hudi_mor/dt=2021-08-28');
```

Response

You can view query results on the client.

13.8.4.14 RUN_TABLE_SERVICE

NOTE

This topic is available for MRS 3.3.1-LTS or later only.

Function

This command is used to perform one-click Compaction/Clean/Archive on Hudi MOR tables. Compaction can be executed only on existing schemas.

Executing a Table Service Command for a Single Table

- **Command format**

  call run_table_service(table => 'table', clean_hours_retained => 'clean_hours_retained', archive_hours_retained => 'archive_hours_retained')

- **Description**

Table 13-37 Parameters

Parameter	Description	Mandatory
table	Table name	Yes
clean_policy	Execution policy used for cleaning.	No. The time policy is used by default.
clean_trigger_strategy	Cleaning interval.	No. By default, a clean operation is triggered once every **clean_hours_retained** hours.
clean_hours_retained	Time frame within which the data files will be retained	No. The default value is 24 hours.
clean_commits_retained	Number of data files written by commits that are retained during cleaning.	No. The default value is **10**.
archive_policy	Execution policy used for archiving.	No. The time policy is used by default.
archive_hours_retained	How many hours in which the timeline will be archived	No. The default value is 24 hours.
max_commit_to_keep	Maximum number of commits metadata files that can be retained during archiving.	No. The default value is **30**.
min_commit_to_keep	Minimum number of commits metadata files that can be retained during archiving.	No. The default value is **20**.
compact_inline	Whether to enable compaction.	No. The default value is **true**.
schedule_compact_inline	Whether to perform only the schedule operation after compaction is enabled. **CAUTION** If you want the **run_table_service** command to generate a compaction plan and not to execute it, set **schedule_compact_inline** to **true** and **run_compact_inline** to **false**.	No. The default value is **false**.

Parameter	Description	Mandatory
run_compact_inline	Whether to perform only the run operation after compaction is enabled. **CAUTION** By default, the **run_table_service** command executes only the existing compaction plan in the Hudi table. If you want the **run_table_service** command to generate a new compaction plan and execute it, set **schedule_compact_inline** to **false** and **run_compact_inline** to **false**.	No. The default value is **true**.
compact_max_delta_commits	Number of commits that triggers compaction.	No. The default value is **5**.

- **Example**

```
// Clean and archive based on the time policy.
call run_table_service(table => 'hudi_table', clean_hours_retained => 1, archive_hours_retained => 1)

// Clean and archive data based on the commits policy.
call run_table_service(table => 'hudi_table', clean_policy => 'KEEP_LATEST_COMMITS',
clean_trigger_strategy => 'NUM_COMMITS', clean_commits_retained => 2, archive_policy =>
'KEEP_META_FILES_BY_COMMITS', min_commit_to_keep => 3, max_commit_to_keep => 4);
```

- **Response**

  You can view the results on the client.

Concurrently Executing Table Service Commands for Multiple Tables

- **Command format**

  call run_table_service(tables => 'table1;table2;table3', cleanNums => 'clean_commits_retained', archiveNums=> 'archive_commits_retained')

- **Description**

  Table 13-38 Description

Parameter	Description	Mandatory
tables	Names of tables where the table service commands will be executed for. Use semicolons (;) to separate multiple table names.	Yes
cleanNums	Number of commits retained during cleaning.	No

Parameter	Description	Mandatory
archiveNums	Number of commits retained during archiving.	No. If this item is set, the value must be greater than the value of **cleanNums**.

- **Example**

```
// Clean and archive data based on the time policy.
call run_table_service(tables => 'tb1;tb2;tb3', cleanNums  => 10, archiveNums => 12)
```

- **Response**

  You can view the results on the client.

13.8.4.15 SYNC_HIVE

NOTE

This topic is available for MRS 3.5.0-LTS or later only.

Function

This command is used to synchronize Hudi tables stored in the Hudi data directory to Hive.

Syntax

call sync_hive(table => '[table]', tablePath => '[tablePath]')

Description

Table 13-39 Parameters

Parameter	Description	Mandatory
table	Name of the table to be synchronized to Hive.	Yes. If the table does not exist, it will be created.
tablePath	Storage path of the Hudi data directory	Yes

Example

```
call sync_hive(table => 'hudi_table', tablePath => '/user/hive/warehouse/hudi_table')
```

Response

You can view query results on the client.

13.8.5 TTL

13.8.5.1 Overview

> **CAUTION**
>
> The Time To Live (TTL) function is available in MRS 3.5.0 and later versions only.
>
> Only partition-level Time To Live (TTL) is supported.

The TTL mechanism controls the lifecycle of data in the Hudi table, which is also called the process of data aging. If the TTL function is disabled, you may encounter the following scenarios:

- Scenario 1: If you want to delete an aged partition for stream processing, you must stop the stream job and run a DDL command to delete the partition.
- Scenario 2: Aged partitions cannot be deleted in batches. You can use a TTL rule to age data in each partition in streaming jobs or asynchronous jobs.

13.8.5.2 Initializing the Partitions of Inventory Tables

Function

Calculate the last modification time of each historical partition and write the time to the **.hoodie_partition_metadata** file in each partition in the format of **lastUpdateTime=***Last modification time*. TTL determines whether a partition is aged based on the result of Current system time – Last modification time of the partition.

Syntax

```
call ttl_update_partitions(table => "[table]", path => "[path]", dryRun => [dryRun])
```

> **NOTE**
>
> - If **dryRun** is **true**, the last modification time of each partition is printed, but is not written to the **.hoodie_partition_metadata** file.
> - When **dryRun** is **false**, the last modification time of each partition is printed and written to the **.hoodie_partition_metadata** file.

Description

Parameter	Description	Mandatory
table	Table name. *Database name*.*Table name* is also supported. The value is a string.	No. Specify either **table** or **path**.
path	Absolute path of the table. The value is a string.	No. Specify either **table** or **path**.
dryRun	The value is of the Boolean type.	The value can be **false** or **true**. The default value is **false**.

Examples

Initialize TTL by specifying a table name.

```
call ttl_update_partitions(table => "hudi_table", dryRun => true)
```

Initialize TTL by specifying a path.

```
call ttl_update_partitions(path => "hdfs://hacluster/user/hive/warehouse/hudi_table/", dryRun => true)
```

13.8.5.3 Enabling and Disabling TTL

Function

Enable or disable the TTL of the Hudi table.

Syntax

```
call ttl_configuration(table => "[table]", path => "[path]", enabled => "[enabled]", strategy => "[strategy]", value => "[value]", resolveConflictsBy => "[resolveConflictsBy]", runInline => "[runInline]")
```

NOTICE

- If **strategy** is **NUM_COMMITS**, **value** indicates that TTL is triggered every [*value*] commits/deltacommits.
- If **strategy** is **TIME_ELAPSED**, **value** indicates that TTL is triggered every [*value*] days.
- The **resolveConflictsBy** attribute is used to handle TTL policy conflicts. For example, the **dt=2023/05/01** partition is hit by the following two TTL policies:

  Policy A: call ttl_policy_save(table => 'hudi_table', spec => 'dt=2023/05/*', units => 'DAYS', value => '2', level => 'PARTITION');

  Policy B: call ttl_policy_save(table => 'hudi_table', spec => 'dt=2023/05/01', units => 'DAYS', value => '1', level => 'PARTITION');
- If **resolveConflictsBy** is **MAX_TTL** and a partition is hit by multiple TTL policies, the policy with the longest retention period has the highest priority. Policy A takes effect on the **dt=2023/05/01** partition, but policy B does not take effect.
- If **resolveConflictsBy** is **MIN_TTL** and a partition is hit by multiple TTL policies, the policy with the shortest retention period has the highest priority. Policy B takes effect on the **dt=2023/05/01** partition, but policy A does not take effect.
- If **runInline** is **true**, TTL determines whether to check the TTLs of all partitions based on the values of **strategy** and **value** after each write operation is complete. If the TTL needs to be checked and aged partitions are detected, the aged partitions are logically deleted, and the logical deletion operation generates a **repalcecommit**. When the clean operation is performed, the aged partitions are automatically and physically deleted. If the check is not required, the next write operation is performed.
- If **runInline** is **false**, you do not need to specify **strategy** or **value**. TTL checks whether all partitions meet aging conditions during each clean operation. If they do, **aged partitions are physically deleted.**

Description

Parameter	Description	Mandatory
table	Table name. *Database name*.*Table name* is also supported. The value is a string.	No. Specify either **table** or **path**.
path	Absolute path of the table. The value is a string.	No. Specify either **table** or **path**.
enabled	Whether to enable TTL. The value is a string.	No. The value can be **false** or **true**. The default value is **false**.
strategy	How TTL is defined. The value is a string.	No. The value can be **NUM_COMMITS** or **TIME_ELAPSED**. The default value is **NUM_COMMITS**.
value	TTL triggering interval. The value is a string.	No. The default value is **10**.
resolveConflictsBy	Policy to handle TTL conflicts. The value is a string.	No. The value can be **MAX_TTL** or **MIN_TTL**. The default value is **MAX_TTL**.
runInline	Whether the TTL event is triggered when the write operation is complete. The value is a string.	No. The value can be **false** or **true**. The default value is **false**.

Examples

- Enable TTL for a specified table. The TTL event is triggered every 10 replacecommits.
  ```
  call ttl_configuration(table => "hudi_table", enabled => "true", strategy => "NUM_COMMITS", value => "10", resolveConflictsBy => "MAX_TTL", runInline => "true")
  ```
- Enable TTL for a specified path. The TTL event is triggered once every day.
  ```
  call ttl_configuration(path => "hdfs://hacluster/user/hive/warehouse/hudi_table/", enabled => "true", strategy => "TIME_ELAPSED", value => "1", resolveConflictsBy => "MIN_TTL", runInline => "true")
  ```
- Enable TTL for a specified table. The TTL event is triggered each time clean is performed.
  ```
  call ttl_configuration(table => "hudi_table", enabled => "true", resolveConflictsBy => "MAX_TTL", runInline => "false")
  ```
- Disable TTL.
  ```
  call ttl_configuration(table => "hudi_table", enabled => "false")
  ```


13.8.5.4 Adding/Updating/Deleting/Clearing/Viewing TTL Policies

Function

Add, update, delete, and clear TTL policies.

Syntax

- Add a TTL policy.
  ```
  call ttl_policy_save(table => "[table]", path => "[path]", spec => "[spec]", level => "[level]", value => "[value]", units => "[units]")
  ```

  NOTE

  spec uses regular expressions to select partitions that require TTL. You can run the **show partitions** command to view the partition format of the Hudi table and write a correct regular expression to match partitions.

  value and **units** must be used together. **value** is an integer, and **units** is the unit of the integer value, indicating the duration for storing data.
- Update TTL policy.
  ```
  call ttl_policy_save(table => "[table]", path => "[path]", spec => "[spec]", level => "[level]", value => "[value]", units => "[units]")
  ```

  NOTE

  If **spec** has been specified (same as that of an existing policy), the historical TTL policy is updated.
- Delete a TTL policy.
  ```
  call ttl_policy_delete(table => "[table]", path => "[path]", spec => "[spec]")
  ```
- Clear a TTL policy.
  ```
  call ttl_policy_empty(table => "[table]", path => "[path]")
  ```
- View a TTL policy.
  ```
  call ttl_policy_show(table => "[table]", path => "[path]")
  ```

Parameters

Parameters of the **ttl_policy_save** command are listed in the following table.

Parameter	Description	Mandatory
table	Table name. *Database name*.*Table name* is also supported. The value is a string.	No. Specify either **table** or **path**.
path	Absolute path of the table. The value is a string.	No. Specify either **table** or **path**.
spec	Regular expression. The value is a string.	Yes
level	TTL level. The value is a string.	Yes. The value can be **PARTITION** or **RECORD**. Currently, only **PARTITION** is supported.

Parameter	Description	Mandatory
value	Retention duration. The value is a string.	Yes
units	Unit of the retention duration. The value is a string.	Yes. The value can be **YEARS**, **MONTHS**, **WEEKS**, and **DAYS**.

The parameters of the **ttl_policy_delete** command are listed in the following table.

Parameter	Description	Mandatory
table	Table name. *Database name*.*Table name* is also supported. The value is a string.	No. Specify either **table** or **path**.
path	Absolute path of the table. The value is a string.	No. Specify either **table** or **path**.
spec	Regular expression. The value is a string.	Yes

The parameters of the **ttl_policy_empty** command are listed in the following table.

Parameter	Description	Mandatory
table	Table name. *Database name*.*Table name* is also supported. The value is a string.	No. Specify either **table** or **path**.
path	Absolute path of the table. The value is a string.	No. Specify either **table** or **path**.

The parameters of the **ttl_policy_show** command are listed in the following table.

Parameter	Description	Mandatory
table	Table name. *Database name*.*Table name* is also supported. The value is a string.	No. Specify either **table** or **path**.

Parameter	Description	Mandatory
path	Absolute path of the table. The value is a string.	No. Specify either **table** or **path**.

Examples

Scenario: The Hudi table has three levels of partitions: year, month, and day, and the partition field is **dt**.

- Add a TTL policy to retain the partition of May 2023 for one day.
  ```
  call ttl_policy_save(table => 'hudi_table', spec => 'dt=2023/05/*', units => 'DAYS', value => '1', level => 'PARTITION')
  ```
- Update the TTL policy to retain the partition of May 2023 for one day.
  ```
  call ttl_policy_save(table => 'hudi_table', spec => 'dt=2023/05/*', units => 'YEARS', value => '1', level => 'PARTITION')
  ```
- Delete the TTL policy.
  ```
  call ttl_policy_delete(table => "hudi_table", spec => "dt=2023/05/*")
  ```
- Clear the TTL policy.
  ```
  call ttl_policy_empty(table => "hudi_table")
  ```
- View the TTL policy.
  ```
  call ttl_policy_show(table => "hudi_table")
  ```

13.8.5.5 Triggering TTL Events

Function

A TTL event is triggered based on the interval configured using **ttl_configuration**. However, you can make the TTL go off immediately.

Syntax

```
call ttl_policy_run(table => "[table]", path => "[path]", dryRun => [dryRun])
```

NOTE

- If **dryRun** is **true**, aged partitions are detected based on the TTL policy configured in the Hudi table and only aged partitions are printed.
- If **dryRun** is **true**, aged partitions are detected based on the TTL policy configured in the Hudi table and only aged partitions are logically deleted. **The logical deletion operation generates a repalcecommit. When the clean operation is performed, the aged partitions are automatically and physically deleted.**

Parameters

Parameter	Description	Mandatory
table	Table name. *Database name*.*Table name* is also supported. The value is a string.	No. Specify either **table** or **path**.

Parameter	Description	Mandatory
path	Absolute path of the table. The value is a string.	No. Specify either **table** or **path**.
dryRun	The value is of the Boolean type.	No. The value can be **false** or **true**. The default value is **false**.

Examples

Make a TTL go off and execute the deletion immediately.

```
call ttl_policy_run(table => "hudi_table", dryRun => false)
```

13.9 Hudi Schema Evolution

13.9.1 Evolution Introduction

Schema evolution allows users to easily change the current schema of a Hudi table to adapt to the data that is changing over time.

NOTE

This topic is available for MRS 3.2.0 or later versions only.

Schema Evolution Scenarios

Schema evolution scenarios

- Columns (including nested columns) can be added, deleted, modified, and moved.
- Partition columns cannot be evolved.
- You cannot add, delete, or perform operations on nested columns of the Array type.

Table 13-40 Engines supported

Component	DDL Operation	Hudi Table Write Operation	Hudi Table Read Operation	Hudi Table Compaction Operation
SparkSQL	Y	Y	Y	Y
Flink	N	Y	Y	Y
HetuEngine	N	N	Y	N
Hive	N	N	Y	N

13.9.2 Configuring SparkSQL for Hudi Schema Evolution

CAUTION

- Schema evolution cannot be disabled once being enabled.
- This topic is available for MRS 3.2.0 and earlier versions only.

- To use spark-beeline, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark2x**, and click the **Configurations** tab then the **All Configurations** sub-tab.

  Search for **spark.sql.extensions** in the search box and change its value of JDBCServer to **org.apache.spark.sql.hive.FISparkSessionExtension,org.apache.spark.sql.hudi.HoodieSparkSessionExtension,org.apache.spark.sql.hive.CarbonInternalExtensions**.
- For SQL operations, run the following command before running any SQL statements:
  ```
  set hoodie.schema.evolution.enable=true
  ```
- For API calls, specify the following parameter in DataFrame options:
  ```
  hoodie.schema.evolution.enable -> true
  ```

13.9.3 Hudi Schema Evolution and Syntax

13.9.3.1 ADD COLUMNS

Function

The **ADD COLUMNS** command is used to add a column to an existing table.

Syntax

ALTER TABLE *Table name* **ADD COLUMNS***(col_spec[, col_spec ...])*

Parameter Description

Table 13-41 ADD COLUMNS parameters

Parameter	Description
tableName	Table name.

Parameter	Description
col_spec	Column specifications, consisting of five fields, **col_name**, **col_type**, **nullable**, **comment**, and **col_position**. • **col_name**: name of the new column. It is mandatory. To add a sub-column to a nested column, specify the full name of the sub-column in this field. For example: – To add sub-column **col1** to a nested struct type column **column users struct<name: string, age: int>**, set this field to **users.col1**. – To add sub-column **col1** to a nested map type column **member map<string, struct<n: string, a: int>>**, set this field to **member.value.col1**. – To add sub-column **col2** to a nested array type column **arraylike array<struct<a1: string, a2: int>>**, set this field to **arraylike.element.col2**. • **col_type**: type of the new column. It is mandatory. • **nullable**: whether the new column can be null. The value can be left empty. • **comment**: comment of the new column. The value can be left empty. • **col_position**: position where the new column is added. The value can be **FIRST** or **AFTER origin_col**. If it is set to **FIRST**, the new column will be added to the first column of the table. If it is set to **AFTER origin_col**, the new column will be added after original column **origin_col**. The value can be left empty. **FIRST** can be used only when new sub-columns are added to nested columns. Do not use **FIRST** in top-level columns. There are no restrictions about the usage of **AFTER**.

Example

```
alter table h0 add columns(ext0 string);
alter table h0 add columns(new_col int not null comment 'add new column' after col1);
alter table complex_table add columns(col_struct.col_name string comment 'add new column to a struct col' after col_from_col_struct);
```

Response

You can run the **DESCRIBE** command to view the new column.

13.9.3.2 ALTER COLUMN

Function

The **ALTER TABLE ... ALTER COLUMN** command is used to change the attributes of a column, such as the column type, position, and comment.

Syntax

ALTER TABLE *Table name* **ALTER**

[COLUMN] *col_old_name* **TYPE** *column_type*

[COMMENT] *col_comment*

[FIRST|AFTER] *column_name*

Parameter Description

Table 13-42 ALTER COLUMN parameters

Parameter	Description
tableName	Table name.
col_old_name	Name of the column to be altered.
column_type	Type of the target column.
col_comment	Column comment.
column_name	New position to place the target column. For example, **AFTER column_name** indicates that the target column is placed after **column_name**.

Example

- Changing the column type

```
ALTER TABLE table1 ALTER COLUMN a.b.c TYPE bigint
```

  a.b.c indicates the full path of a nested column. For details about the nested column rules, see **ADD COLUMNS**.

  The following changes on column types are supported:

  - int => long/float/double/string/decimal
  - long => float/double/string/decimal
  - float => double/String/decimal
  - From double to string or decimal
  - From decimal to decimal or string
  - From string to date or decimal
  - From date to string

- Altering other attributes


```
ALTER TABLE table1 ALTER COLUMN a.b.c DROP NOT NULL
ALTER TABLE table1 ALTER COLUMN a.b.c COMMENT 'new comment'
ALTER TABLE table1 ALTER COLUMN a.b.c FIRST
ALTER TABLE table1 ALTER COLUMN a.b.c AFTER x
```

a.b.c indicates the full path of a nested column. For details about the nested column rules, see **ADD COLUMNS**.

Response

You can run the **DESCRIBE** command to view the modified column.

13.9.3.3 DROP COLUMN

Function

The **ALTER TABLE ... DROP COLUMN** command is used to delete a column.

Syntax

ALTER TABLE *tableName* **DROP COLUMN|COLUMNS** *cols*

Parameter Description

Table 13-43 DROP COLUMN parameters

Parameter	Description
tableName	Table name.
cols	Columns to be deleted. You can specify multiple columns.

Example

```
ALTER TABLE table1 DROP COLUMN a.b.c
ALTER TABLE table1 DROP COLUMNS a.b.c, x, y
```

a.b.c indicates the full path of a nested column. For details about the nested column rules, see **ADD COLUMNS**.

Response

You can run the **DESCRIBE** command to check which column is deleted.

13.9.3.4 RENAME

Function

The **ALTER TABLE ... RENAME** command is used to change the table name.

Syntax

ALTER TABLE *tableName* **RENAME TO** *newTableName*

Parameter Description

Table 13-44 RENAME parameters

Parameter	Description
tableName	Table name.
newTableName	New table name.

Example

```
ALTER TABLE table1 RENAME TO table2
```

Response

You can run the **SHOW TABLES** command to view the new table name.

13.9.3.5 SET

Function

The **ALTER TABLE ... SET|UNSET** command is used to modify table properties.

Syntax

ALTER TABLE *Table name* **SET|UNSET** *tblproperties*

Parameter Description

Table 13-45 SET|UNSET parameters

Parameter	Description
tableName	Table name.
tblproperties	Table properties.

Example

```
ALTER TABLE table SET TBLPROPERTIES ('table_property' = 'property_value')
ALTER TABLE table UNSET TBLPROPERTIES [IF EXISTS] ('comment', 'key')
```

Response

You can run the **DESCRIBE** command to view new table properties.

13.9.3.6 RENAME COLUMN

Function

The **ALTER TABLE ... RENAME COLUMN** command is used to change the column name.

Syntax

ALTER TABLE *tableName* **RENAME COLUMN** *old_columnName* **TO** *new_columnName*

Parameter Description

Table 13-46 RENAME COLUMN parameters

Parameter	Description
tableName	Table name.
old_columnName	Old column name.
new_columnName	New column name.

Example

```
ALTER TABLE table1 RENAME COLUMN a.b.c TO x
```

a.b.c indicates the full path of a nested column. For details about the nested column rules, see **ADD COLUMNS**.

NOTE

After the column name is changed, the change is automatically synchronized to the column comment. The comment is in **rename oldName to newName** format.

Response

You can run the **DESCRIBE** command to view the new column name.

13.9.4 Concurrency in the Hudi Schema Evolution

CAUTION

When creating a table, you need to set **hoodie.cleaner.policy.failed.writes** to **LAZY**. Otherwise, rollback will be triggered when concurrent submission operations are performed.

DDL Concurrency

Table 13-47 Concurrent DDL operations

DDL Operation	add	rename	change type	change comment	drop
add	Y	Y	Y	Y	Y
rename	Y	Y	Y	Y	Y
change type	Y	Y	Y	Y	Y
change comment	Y	Y	Y	Y	Y
drop	Y	Y	Y	Y	N

NOTE

When performing DDL operations on the same column concurrently, pay attention to the following:

- Multiple drop operations cannot be concurrently performed at a time on the same column. Only the first drop operation can be successfully performed and then exception message "java.lang.UnsupportedOperationException: cannot evolution schema implicitly, the column for which the update operation is performed does not exist." is thrown.
- When drop, rename, change type, and change comment operations are concurrently performed, drop operations must be executed the last. Otherwise, only drop and operations before drop can be performed, and exception message "java.lang.UnsupportedOperationException: cannot evolution schema implicitly, the column for which the update operation is performed does not exist." is reported when operations after drop are performed.

DDL and DML Concurrency

Table 13-48 Concurrent DDL and DML operations

DDL Operation	insert into	update	delete	set/reset
add	Y	Y	Y	Y
rename	N	N	Y	N
change type	N	N	Y	N
change comment	Y	Y	Y	Y
drop	N	N	Y	N

NOTE

Exception message "cannot evolution schema implicitly, actions such as rename, delete, and type change were found" is thrown when unsupported DDL or DML operations are performed concurrently.

13.10 Configuring Default Values for Hudi Data Columns

This feature allows you to set default values for columns when you add columns to a table. When you query historical data, the default value is returned for the new column.

NOTE

This topic is available for MRS 3.3.0 or later only.

Constraints

- If data has been rewritten before default values are set for a new column, the default values of the column cannot be returned when historical data is queried. In this case, NULL values are returned. Some or all data will be rewritten when data is imported to the database, updated, compacted, or clustered.
- The default values of a column must match the column type. If they do not match, the type will be forcibly converted. This loses the precision of the default values or changes the default values to NULL.
- The default values of historical data are the default values set for the column for the first time. Changing the default values of a column for multiple times does not affect the query result of historical data.
- After the default value is set, it cannot be rolled back.
- Currently, Spark SQL does not support the function of viewing default column values. You can run the **show create table** command on Hive beeline to view default column values.

Scope

Currently, only the **int**, **bigint**, **float**, **double**, **decimal**, **string**, **date**, **timestamp**, **boolean**, and **binary** data types are supported.

Table 13-49 Engines supported

Component	DDL Operation	Support for Write Operation	Support for Read Operation
SparkSQL	Y	Y	Y
Spark DataSource	N	N	Y
Flink	N	N	Y
HetuEngine	N	N	Y

Component	DDL Operation	Support for Write Operation	Support for Read Operation
Hive	N	N	Y

Example

For details about the SQL syntax, see **Hudi SQL Syntax Reference**.

The following is an example:

- Create a table and specify default values for columns.
```
create table if not exists h3(
id bigint,
name string,
price double default 12.34
) using hudi
options (
primaryKey = 'id',
type = 'mor',
preCombineField = 'name'
);
```
- Add columns and specify default values for the columns.
```
alter table h3 add columns(col1 string default 'col1_value');
alter table h3 add columns(col2 string default 'col2_value', col3 int default 1);
```
- Change default values of columns.
```
alter table h3 alter column price set default 14.56;
```
- Inset data and use column default values.
```
insert into h3(id, name) values(1, 'aaa');
insert into h3(id, name, price) select 2, 'bbb', 12.5;
```

13.11 Partial Update

NOTE

This section applies only to MRS 3.3.1-LTS and later.

Hudi allows users to update columns partially. You can use the latest data under the same primary key to update fields in different columns of each row one by one until the entire data is complete.

Scenarios

Currently, the open source community provides PartialUpdateAvroPayload to update columns partially. However, when multiple streams are updated and different columns are updated in each stream, data is mutually overwritten.

By introducing the sequence group, Hudi can solve this problem and implement real partial update.

Columns in a table are divided into different sequence groups as you need during table creation. Whether the columns in each sequence group are updated is determined by the **precombine** field of the sequence group. The sequence groups are not affected by each other.

Constraints

- Due to the restrictions of Hudi OCC, you are not advised to concurrently write data to a Hudi table in multiple streams. If multiple streams need to be written at the same time, union all streams before write them to Hudi.
- New columns can be added and grouped. To add the column to a new group, you need to modify **tblproperties** and **serdeproperties** of the table. Example statements are as follows:
  - Add new columns **col5,col6,group_3**.

    alter table testTable add columns (col5 int, col6 int, group_3 int);
  - Add the information of the new group to **tblproperties**.

    alter table testTable set tblproperties('fields.group_3.sequence-group' = 'col5,col6');
  - Add the information of the new group to **serdeproperties**.

    alter table testTable set serdeproperties('fields.group_3.sequence-group' = 'col5,col6');
- Columns in a sequence group cannot overlap. For example, there cannot be the same **col1** column in both **sequence-1** and **sequence-2**.
- For the **group** column, only int, bigint, float, double, date, and timestamp types are supported.
- If partial update is enabled for existing tables, you must operate in compliance with the following requirements. Otherwise, the data may not meet the expectation.
  - Stop writing data to the table to be modified.
  - Full compaction must be performed for the MOR table.

    Forcibly enable compaction.

    set hoodie.compaction.inline.max.delta.commits=1;

    set hoodie.compact.inline=true;

    Perform full compaction.

    run compaction on my_table;

    reset hoodie.compaction.inline.max.delta.commits;
  - Add the information of the new group to **tblproperties**.

    alter table testTable set tblproperties('fields.group_1.sequence-group' = 'col1,col2');
  - Add the information of the new group to **serdeproperties**.

    alter table testTable set serdeproperties('fields.group_1.sequence-group' = 'col1,col2');

Example

```
create table if not exists testTable(
id INT,
col1 INT,
col2 INT,
group_1 INT,
col3 INT,
col4 INT,
group_2 INT,
ts LONG
```



```
) using hudi
tblproperties (
primaryKey = 'id',
type = 'mor',
'hoodie.merge-engine' = 'partial-update',
'fields.group_1.sequence-group' = 'col1,col2',---- Divide update groups. Whether the col1 and col2 columns are updated is determined by group_1. That is, group_1 is the preCombine field of the group.
'fields.group_2.sequence-group' = 'col3,col4' ---- Divide update groups. Whether the col3 and col4 columns are updated is determined by group_2. That is, group_2 is the preCombine field of the group.
);
---Run the insert statement.
insert into testTable values (1, 1, 1, 1, 1, 1, 1000); -- First write
insert into testTable values (1, 2, 2, 2, 2, null, 2000); -- Second write
---Query result
select * from testTable;
---The result is 1, 2, 2, 2, 1, 1, 1, 2000.
Result description: When data is written for the second time, the value of group_1 is 2, which is greater than the historical value 1 in the table. This means that col1 and col2 are updated. The value of group_2 is null, which means that col3 and col4 are not updated.
---Execute the insert statement.
insert into testTable values (1, null, 3, 2, 3, 3, 3, 4000);  -- Third write
select * from testTable;
---The result is 1, null, 3, 2, 3, 3, 3, 4000.
---Result description: When data is written for the third time, the value of group_1 is 2, which is greater than or equal to historical value 2 in the table. This means that, col1 and col2 are updated. The value of group_2 is 3, which is greater than historical value 1 in the table. This means that col3 and col4 are updated.
```

Application Scenario Example

The union write simulates a multi-table join. (Flink can use this function to join streams.)

The following code uses SparkSQL as an example. Hudi is used to convert the join of **t1** and **t2** to a union and an insert operation to reduce join overhead.

```
create table if not exists t1(
id INT,
col1 INT,
col2 INT
) using parquet;
insert into t1 values(1, 1, 1);
create table if not exists t2(
id INT,
col3 INT,
col4 INT
) using parquet;
insert into t2 values(1, 2, 2);
create table if not exists joinTable(
id INT,
col1 INT,
col2 INT,
group_1 INT,
col3 INT,
col4 INT,
group_2 INT
) using hudi
tblproperties (
primaryKey = 'id',
type = 'mor',
'hoodie.index.type' = 'BUCKET',
hoodie.bucket.index.num.buckets=1,
'hoodie.merge-engine' = 'partial-update',
'fields.group_1.sequence-group' = 'col1,col2',
'fields.group_2.sequence-group' = 'col3,col4'
);
```



```
--- Union and insert are performed to simulate the join operation.
insert into joinTable
select id, col1, col2, 1, null, null, null from t1      --- Use any non-null value as the precombine of col1 and col2.
union all
select id, null, null, null, col3, col4, 1 from t2;      --- Use any non-null value as the precombine of col3 and col4.

---Execute the query.
select id,col1,col2,col3,col4 from joinTable;
-- Result
-- 1, 1, 1, 2, 2
```

13.12 Aggregate Functions in Hudi

NOTE

This section is available for MRS 3.5.0-LTS and later versions only.

Scenarios

The open-source community provides a pluggable payload mechanism to meet various aggregation requirements. To simplify payload development, MRS offers built-in typical aggregation functions. You can use these functions provided by Hudi to aggregate data with the same primary key.

Currently, the following aggregate functions and data types are supported:

- sum: aggregates values across rows. It supports the following data types: DECIMAL (decimal), SHORT (small integer), INTEGER (integer), BIGINT (large integer), FLOAT (floating point number), and DOUBLE (double-precision floating point number).
- product: calculates the product of rows. It supports the following data types: DECIMAL, SHORT, INTEGER, BIGINT, FLOAT, and DOUBLE.
- count: counts values across rows. The INTEGER and BIGINT data types are supported.
- max: identifies and retains the maximum value. It supports the following data types: STRING, DECIMAL, SHORT, INTEGER, BIGINT, FLOAT, DOUBLE, DATE and TIMESTAMP.
- min: identifies and retains the minimum value. It supports the following data types: STRING, DECIMAL, SHORT, INTEGER, BIGINT, FLOAT, DOUBLE, DATE and TIMESTAMP.
- last_value: replaces values with the latest imported values. All data types are supported.
- last_non_null_value: replaces the values with the latest non-null values. All data types are supported.
- first_value: retrieves the first null value in a dataset. All data types are supported.
- first_non_null_value: selects the first non-null value in the dataset. All data types are supported.

Constraints

- Due to the restrictions of Hudi OCC, you are not advised to concurrently write data to a Hudi table in multiple streams. If multiple streams need to be written at the same time, union all streams before write them to Hudi.
- The functions are suitable for batch read.

Enabling the Aggregation Engine

Set **hoodie.merge-engine=aggregate** in the table creation property to enable the aggregation engine of the Hudi table. The aggregation engine gives each non-primary key field an aggregation function. You can use **fields.<*field-name*>.aggregate-function** to specify the field. The following is an example:

```
create table if not exists testTable(
id INT,
col1 INT,
col2 INT,
col3 INT,
col4 INT,
ts LONG
) using hudi
tblproperties (
primaryKey = 'id',
preCombineField='ts'
type = 'mor',
'hoodie.merge-engine' = 'aggregate',
'fields.col2.aggregate-function' = 'count', ----- The count operation is performed for col2 after aggregation based on the primary key.
'fields.col3.aggregate-function' = 'max' ----- The max operation is performed for col3 after aggregation based on the primary key.
);
```

13.13 Typical Hudi Configuration Parameters

This section describes important Hudi configurations. For details, visit the Hudi official website at **https://hudi.apache.org/docs/configurations/**.

Write Configuration

Table 13-50 Write configuration parameters

Parameter	Description	Default Value
hoodie.datasource.write.table.name	Name of the Hudi table to which data is written	None

Parameter	Description	Default Value
hoodie.datasource.write.operation	Type of the operation for writing data to the Hudi table. Value options are as follows: • **upsert**: updates and inserts data. • **delete**: deletes data. • **insert**: inserts data. • **bulk_insert**: imports data during initial table creation. Do not use **upsert** or **insert** during initial table creation. • **insert_overwrite**: performs insert and overwrite operations on static partitions. • **insert_overwrite_table**: performs insert and overwrite operations on dynamic partitions. It does not immediately delete the entire table or overwrite the table. Instead, it overwrites the metadata of the Hudi table logically, and Hudi deletes useless data through the clean mechanism. Its efficiency is higher than that of the sum of **bulk_insert** and **overwrite**.	upsert
hoodie.datasource.write.table.type	Type of the Hudi table. This parameter cannot be modified once specified. The value can be **MERGE_ON_READ**.	COPY_ON_WRITE
hoodie.datasource.write.precombine.field	Merges and reduplicates rows with the same key before write.	A specific table field
hoodie.datasource.write.payload.class	Class used to merge the records to be updated and the updated records during update. This parameter can be customized. You can compile it to implement your merge logic.	org.apache.hudi.common.model.DefaultHoodieRecordPayload
hoodie.datasource.write.recordkey.field	Unique primary key of the Hudi table	A specific table field

Parameter	Description	Default Value
hoodie.datasource.write.partitionpath.field	Partition key. This parameter can be used together with **hoodie.datasource.write.keygenerator.class** to meet different partition needs.	None
hoodie.datasource.write.hive_style_partitioning	Whether to specify a partition mode that is the same as that of Hive. Set this parameter to **true**.	true
hoodie.datasource.write.keygenerator.class	Used with **hoodie.datasource.write.partitionpath.field** and **hoodie.datasource.write.recordkey.field** to generate the primary key and partition mode. **NOTE** If the value of this parameter is different from that saved in the table, a message is displayed, indicating that the value must be the same.	org.apache.hudi.keygen.ComplexKeyGenerator

Configuration of Hive Table Synchronization

Table 13-51 Parameters for synchronizing Hive tables

Parameter	Description	Default Value
hoodie.datasource.hive_sync.enable	Whether to synchronize Hudi tables to Hive MetaStore. **CAUTION** Set this parameter to **true** to use Hive to centrally manage Hudi tables.	false
hoodie.datasource.hive_sync.database	Name of the database to be synchronized to Hive	default
hoodie.datasource.hive_sync.table	Name of the table to be synchronized to Hive. Set this parameter to the value of **hoodie.datasource.write.table.name**.	unknown
hoodie.datasource.hive_sync.username	Username used for Hive synchronization	hive

Parameter	Description	Default Value
hoodie.datasource.hive_sync.password	Password used for Hive synchronization	hive
hoodie.datasource.hive_sync.jdbcurl	Specified connection to the Hive JDBC	""
hoodie.datasource.hive_sync.use_jdbc	Whether to use Hive JDBC to connect to Hive and synchronize Hudi table information. Set this parameter to **false** to invalidate the JDBC connection configuration.	true
hoodie.datasource.hive_sync.partition_fields	Hive partition columns	""
hoodie.datasource.hive_sync.partition_extractor_class	Class used to extract Hudi partition column values and convert them into Hive partition columns.	org.apache.hudi.hive.SlashEncodedDayPartitionValueExtractor
hoodie.datasource.hive_sync.support_timestamp	If the Hudi table contains a field of the timestamp type, set this parameter to **true** to synchronize the timestamp type to the Hive metadata. The default value is **false**, indicating that the timestamp type is converted to bigint during synchronization by default. In this case, an error may occur when you query a Hudi table that contains a field of the timestamp type using SQL statements.	true

Index Configuration

Table 13-52 Index parameters

Parameter	Description	Default Value
hoodie.index.class	Full path of a user-defined index, which must be a subclass of HoodieIndex. When this parameter is specified, the configuration takes precedence over that of **hoodie.index.type**.	""
hoodie.index.type	Index type. The default value is **BLOOM**. The possible options are **BLOOM**, **HBASE**, **GLOBAL_BLOOM**, **SIMPLE**, and **GLOBAL_SIMPLE**. The Bloom filter eliminates the dependency on an external system and is stored in the footer of a Parquet data file.	BLOOM
hoodie.index.bloom.num_entries	This is the number of entries to be stored in the bloom filter. Assume the **maxParquetFileSize** is 128 MB and **averageRecordSize** is 1024 bytes and hence a total of 130 KB records are recorded in a file. The default (60000) is roughly half of this approximation. **CAUTION** Setting this very low will generate many false positives and index lookup will have to scan a lot more files than it has to, and setting this to a very high number will increase the size every data file linearly (roughly 4 KB for every 50,000 entries).	60000
hoodie.index.bloom.fpp	Error rate allowed given the number of entries. This is used to calculate how many bits should be assigned for the bloom filter and the number of hash functions. This is usually set very low (default: **0.000000001**) to tradeoff disk space for lower false positives.	0.000000001

Parameter	Description	Default Value
hoodie.bloom.index.parallelism	Parallelism for index lookup, which involves Spark shuffling. By default, it is automatically calculated based on input workload characteristics.	0
hoodie.bloom.index.prune.by.ranges	When this is set to **true**, range information from files is used to speed up index lookups. This is particularly helpful if the keys have monotonously increasing prefixes, such as timestamp.	true
hoodie.bloom.index.use.caching	When this is set to **true**, the input RDD will be cached to speed up index lookup by reducing I/O for computing parallelism or affected partitions.	true
hoodie.bloom.index.use.treebased.filter	When this is set to **true**, interval tree based file pruning optimization is enabled. This mode speeds up file pruning based on key ranges when compared with the brute-force mode.	true
hoodie.bloom.index.bucketized.checking	When this is set to **true**, bucketized bloom filtering is enabled. This reduces skew seen in sort based bloom index lookup.	true
hoodie.bloom.index.keys.per.bucket	This parameter is available only if **bloomIndexBucketizedChecking** is enabled and the index type is **BLOOM**. This configuration controls the "bucket" size which tracks the number of record-key checks made against a single file and is the unit of work allocated to each partition performing bloom filter lookup. A higher value would amortize the fixed cost of reading the Bloom filter to memory.	10000000

Parameter	Description	Default Value
hoodie.bloom.index.update.partition.path	This parameter is applicable only when the index type is **GLOBAL_BLOOM**. If this parameter is set to **true**, an update including the partition path of a record that already exists will result in the insertion of the incoming record into the new partition and the deletion of the original record in the old partition. If this parameter is set to **false**, the original record will only be updated in the old partition.	true
hoodie.index.hbase.zkquorum	Mandatory. This parameter is available only when the index type is **HBase**. HBase ZooKeeper quorum URL to be connected.	None
hoodie.index.hbase.zkport	Mandatory. This parameter is available only when the index type is **HBase**. HBase ZooKeeper quorum port to be connected.	None
hoodie.index.hbase.zknode.path	Mandatory. This parameter is available only when the index type is **HBase**. It is the root znode that will contain all the znodes created and used by HBase.	None
hoodie.index.hbase.table	Mandatory. This parameter is available only when the index type is **HBase**. HBase table name to be used as an index. Hudi stores the **row_key** and **[partition_path, fileID, commitTime]** mapping in the table.	None

Storage Configuration

Table 13-53 Storage parameter configuration

Parameter	Description	Default Value
hoodie.parquet.max.file.size	Specifies the target size for Parquet files generated in Hudi write phases. For DFS, this parameter needs to be aligned with the underlying file system block size for optimal performance.	120 * 1024 * 1024 byte
hoodie.parquet.block.size	Specifies the Parquet page size. Page is the unit of read in a Parquet file. In a block, pages are compressed separately.	120 * 1024 * 1024 byte
hoodie.parquet.compression.ratio	Specifies the expected compression ratio of Parquet data when Hudi attempts to adjust the size of a new Parquet file. If the size of the file generated by **bulk_insert** is smaller than the expected size, increase the value.	0.1
hoodie.parquet.compression.codec	Specifies the name of the Parquet compression encoding or decoding mode. The default value is **gzip**. Possible options are **[gzip | snappy | uncompressed | lzo]**.	snappy
hoodie.logfile.max.size	Specifies the maximum size of LogFile. It is the maximum size allowed for a log file before it is rolled over to the next version.	1GB
hoodie.logfile.data.block.max.size	Specifies the maximum size of a LogFile data block. It is the maximum size allowed for a single data block to be appended to a log file. It helps to ensure that the data appended to the log file is broken up into sizable blocks to prevent OOM errors. The size should be greater than the JVM memory.	256MB

Parameter	Description	Default Value
hoodie.logfile.to.parquet.compression.ratio	Specifies the expected additional compression when records move from log files to Parquet files. It is used for MOR tables to send inserted content into log files and control the size of compacted Parquet files.	0.35

Compaction and Cleaning Configurations

Table 13-54 Compaction and cleaning parameters

Parameter	Description	Default Value
hoodie.clean.automatic	Whether to perform automatic cleanup.	true
hoodie.cleaner.policy	Cleanup policy to be used. Hudi will delete the Parquet file of an old version to reclaim space. Any query or computation referring to this version of the file will fail. You are advised to ensure that the data retention time exceeds the maximum query execution time.	KEEP_LATEST_COMMITS
hoodie.cleaner.commits.retained	Number of commits to retain. Data will be retained for **num_of_commits * time_between_commits** (scheduled). This also directly translates into the number of datasets can be incrementally pulled.	10
hoodie.keep.max.commits	Number of commits that triggers the archiving operation.	30
hoodie.keep.min.commits	Number of commits reserved for archiving operations.	20
hoodie.commits.archival.batch	Number of commit instants read in memory as a batch and archived together.	10

Parameter	Description	Default Value
hoodie.parquet.small.file.limit	The value must be smaller than that of **maxFileSize**. If **maxFileSize** is set to **0**, this function is disabled. Small files always exist because of the large number of insert records in a partition of batch processing. Hudi provides an option to solve the problem of small files by masking inserts into this partition as updates to existing small files. The size here is the minimum file size that is considered as a "small file size".	104857600 byte
hoodie.copyonwrite.insert.split.size	Parallelism for inserting and writing data. It is the number of inserts grouped for a single partition. Writing out 100 MB files with at least 1 KB records means 100 KB records exist in each file. Overprovision to 500 KB by default. To improve insert latency, adjust the value to match the number of records in a single file. If it is set to a smaller value, the file size will shrink (especially when **compactionSmallFileSize** is set to **0**).	500000
hoodie.copyonwrite.insert.auto.split	Whether Hudi dynamically computes **insertSplitSize** based on the last 24 commit metadata. This function is disabled by default.	true
hoodie.copyonwrite.record.size.estimate	Average record size. If specified, Hudi will use this parameter and not compute dynamically based on the last 24 commit metadata. There is no default value. This is critical in computing the insert parallelism and packing inserts into small files.	1024

Parameter	Description	Default Value
hoodie.compact.inline	If this parameter is set to **true**, compaction is triggered by the ingestion itself right after a commit or delta commit action as part of **insert**, **upsert**, or **bulk_insert**.	true
hoodie.compact.inline.max.delta.commits	Maximum number of delta commits to be retained before inline compression is triggered.	5
hoodie.compaction.lazy.block.read	When **CompactedLogScanner** merges all log files, this parameter helps to choose whether the logblocks should be read lazily. Set it to **true** to use I/O-intensive lazy block read (low memory usage) or **false** to use memory-intensive immediate block read (high memory usage).	true
hoodie.compaction.reverse.log.read	**HoodieLogFormatReader** reads a log file in the forward direction from **pos=0** to **pos=file_length**. If this parameter is set to **true**, Reader reads a log file in reverse direction from **pos=file_length** to **pos=0**.	false
hoodie.cleaner.parallelism	Increase this parameter if cleaning becomes slow.	200
hoodie.compaction.strategy	Which file groups are selected for compaction during each compaction run. By default, Hudi selects the log file with most accumulated unmerged data.	org.apache.hudi.table.action.compact.strategy. LogFileSizeBasedCompactionStrategy
hoodie.compaction.target.io	Number of MBs to spend during compaction run for **LogFileSizeBasedCompactionStrategy**. This parameter can limit ingestion latency when compaction is run in inline mode.	500 * 1024 MB

Parameter	Description	Default Value
hoodie.compaction.daybased.target.partitions	Used by **org.apache.hudi.io.compact.strategy.DayBasedCompactionStrategy** to denote the number of latest partitions to compact during a compaction run.	10
hoodie.compaction.payload.class	It needs to be same as class used during insert or upsert. Similar to writing, compaction also uses the record payload class to merge records in the log against each other, merge again with the base file, and produce the final record to be written after compaction.	org.apache.hudi.common.model.Defaulthoodierecordpayload
hoodie.schedule.compact.only.inline	Whether to generate only a compression plan during a write operation. This parameter is valid only when **hoodie.compact.inline** is set to **true**.	false
hoodie.run.compact.only.inline	Whether to perform only the compression operation when the **run compaction** command is executed using SQL. If the compression plan does not exist, no action is needed.	false

Single-Table Concurrency Control

Table 13-55 Single-table concurrency control configuration

Parameter	Description	Default Value
hoodie.write.lock.provider	Lock provider. You are advised to set the parameter to **org.apache.hudi.hive.HiveMetastoreBasedLockProvider**.	org.apache.hudi.client.transaction.lock.ZookeeperBasedLockProvider
hoodie.write.lock.hivemetastore.database	Hive database.	None
hoodie.write.lock.hivemetastore.table	Hive table name.	None

Parameter	Description	Default Value
hoodie.write.lock.client.num_retries	Retry times.	10
hoodie.write.lock.client.wait_time_ms_between_retry	Retry interval.	10000
hoodie.write.lock.conflict.resolution.strategy	Lock provider class, which must be a subclass of **ConflictResolutionStrategy**.	org.apache.hudi.client.transaction.SimpleConcurrentFileWritesConflictResolutionStrategy
hoodie.write.lock.zookeeper.base_path	Path for storing ZNodes. The parameter must be the same for all concurrent write configurations of the same table.	None
hoodie.write.lock.zookeeper.lock_key	ZNode name. It is recommended that the ZNode name be the same as the Hudi table name.	None
hoodie.write.lock.zookeeper.connection_timeout_ms	ZooKeeper connection timeout interval.	15000
hoodie.write.lock.zookeeper.port	ZooKeeper port.	None
hoodie.write.lock.zookeeper.url	URL of the ZooKeeper.	None
hoodie.write.lock.zookeeper.session_timeout_ms	ZooKeeper session expiration time.	60000

Clustering Configuration

NOTE

This topic is available for MRS 3.1.3 or later versions only.

Clustering has two strategies: **hoodie.clustering.plan.strategy.class** and **hoodie.clustering.execution.strategy.class**. Typically, if **hoodie.clustering.plan.strategy.class** is set to **SparkRecentDaysClusteringPlanStrategy** or **SparkSizeBasedClusteringPlanStrategy**, **hoodie.clustering.execution.strategy.class** does not need to be specified. However, if **hoodie.clustering.plan.strategy.class** is set to **SparkSingleFileSortPlanStrategy**, **hoodie.clustering.execution.strategy.class** must be set to **SparkSingleFileSortExecutionStrategy**.

Table 13-56 Clustering parameter configuration

Parameter	Description	Default Value
hoodie.clustering.inline	Whether to execute clustering synchronously.	false
hoodie.clustering.inline.max.commits	Number of commits that trigger clustering.	4
hoodie.clustering.async.enabled	Whether to enable asynchronous clustering. **NOTE** This parameter is available in MRS 3.3.0-LTS and later versions only.	false
hoodie.clustering.async.max.commits	Number of commits that trigger clustering during asynchronous execution. **NOTE** This parameter is available in MRS 3.3.0-LTS and later versions only.	4
hoodie.clustering.plan.strategy.target.file.max.bytes	Maximum size of each file after clustering.	1024 * 1024 * 1024 byte
hoodie.clustering.plan.strategy.small.file.limit	Files smaller than the value of this parameter are clustered.	300 * 1024 * 1024 byte
hoodie.clustering.plan.strategy.sort.columns	Column used by clustering for sorting.	None
hoodie.layout.optimize.strategy	Clustering execution policy. The value can be **linear**, **z-order**, or **hilbert**.	linear
hoodie.layout.optimize.enable	This function must be enabled when **z-order** and **hilbert** are used.	false
hoodie.clustering.plan.strategy.class	Strategy class for filtering file groups for clustering. By default, files smaller than the value of **hoodie.clustering.plan.strategy.small.file.limit** are filtered.	org.apache.hudi.client.clustering.plan.strategy.SparkSizeBasedClusteringPlanStrategy

Parameter	Description	Default Value
hoodie.clustering.execution.strategy.class	Strategy class for executing clustering (subclass of **RunClusteringStrategy**), which is used to define the execution mode of a cluster plan. The default classes sort the file groups in the plan by the specified column and meet the configured target file size.	org.apache.hudi.client.clustering.run.strategy.SparkSortAndSizeExecutionStrategy
hoodie.clustering.plan.strategy.max.num.groups	Maximum number of file groups that can be selected during clustering. A larger value indicates a higher concurrency.	30
hoodie.clustering.plan.strategy.max.bytes.per.group	Maximum number of data records in each file group involved in clustering	2 * 1024 * 1024 * 1024 byte

13.14 Hudi Performance Tuning

Performance Tuning Methods

In the current version, Spark is recommended for Hudi write operations. Therefore, the tuning methods of Hudi are similar to those of Spark.

Recommended Resource Configuration

- For MOR tables:

  The essence of MOR tables is to write incremental files, so the tuning is based on the data size (dataSize) of Hudi.

  If dataSize is only several GBs, you are advised to run Spark in single-node mode or run Spark in Yarn mode with only one container allocated.

  Parallelism (**p**) of programs for importing data to the lake: **p** = dataSize/128 MB. The number of cores allocated to programs must be the same as the value of **p**. It is recommended that the ratio of the memory size to the number of cores be greater than 1.5:1. That is, a core is configured with 1.5 GB memory. For off-heap memory, it is recommended that the ratio of the memory size to the number of cores be greater than 0.5:1.

- For COW tables:

  The principle of COW tables is to rewrite the original data. Therefore, dataSize and the number of rewritten files must be considered during tuning. Typically, more cores lead to better performance. The number of cores is directly related to the number of rewritten files. The settings of parallelism (**p**) and memory size are similar to those of MOR tables.

13.15 Common Issues About Hudi

13.15.1 "Parquet/Avro schema" Is Reported When Updated Data Is Written

Question

The following error is reported when data is written:

```
org.apache.parquet.io.InvalidRecordException: Parquet/Avro schema mismatch: Avro field 'col1' not found
```

Answer

You are advised to evolve schemas in backward compatible mode while using Hudi. This error usually occurs when you delete some columns, such as **col1**, in backward incompatible mode and then update **col1** written with the old schema in the Parquet file. In this case, the Parquet file attempts to search for all the current fields in the input record, if **col1** does not exist, the preceding exception is reported.

To solve this problem, create an uber schema using all the schema versions evolved and use this uber schema as the target schema. You can obtain a schema from Hive MetaStore and merge it with the current schema.

13.15.2 UnsupportedOperationException Is Reported When Updated Data Is Written

Question

The following error is reported when data is written:

```
java.lang.UnsupportedOperationException: org.apache.parquet.avro.AvroConverters$FieldIntegerConverter
```

Answer

This error will occur again because schema evolutions are in non-backwards compatible mode. Basically, there is some update U for a record R which is already written to the Hudi dataset in the Parquet file. R contains field F which includes certain data type, that is long. U has the same field F with the int data type. Parquet FS does not support incompatible data type conversions.

For such errors, perform valid data type conversions in the data source where you collect data.

13.15.3 SchemaCompatabilityException Is Reported When Updated Data Is Written

Question

The following error is reported when data is written:


```
org.apache.hudi.exception.SchemaCompatabilityException: Unable to validate the rewritten record <record> against schema <schema>at
org.apache.hudi.common.util.HoodieAvroUtils.rewrite(HoodieAvroUtils.java:215)
```

Answer

This error may occur if a schema contains some **non-nullable** field whose value is not present or is null.

You are advised to evolve schemas in backward compatible mode. Essentially, this means either you need to set each newly added field to null or to default values. In Hudi 0.5.1 and later versions, the troubleshooting is invalid if fields rely on default values.

13.15.4 What Should I Do If Hudi Consumes Much Space in a Temporary Folder During Upsert?

Question

Hudi consumes much space in a temporary folder during upsert.

Answer

Hudi will spill part of input data to disk if the maximum memory for merge is reached when much input data is upserted.

If the memory is sufficient, increase the memory of the Spark executor and add the **hoodie.memory.merge.fraction** option, for example, **option("hoodie.memory.merge.fraction", "0.8")**.

13.15.5 Hudi Fails to Write Decimal Data with Lower Precision

Question

Decimal data is initially written to a Hudi table using the **BULK_INSERT** command. Then when data is subsequently written using **UPSERT**, the following error is reported:

```
java.lang.UnsupportedOperationException: org.apache.parquet.avro.AvroConverters$FieldFixedConverter
```

Answer

Cause:

The Hudi table contains decimal data.

The initial bulk insert of data is implemented using the Spark class for writing Parquet files. However, Spark processes the decimal data with different precisions differently.

When data is written using the **UPSERT** command, Hudi uses the Avro-compliant class for writing Parquet files, which is incompatible with the Spark class.

Solutions:

When executing the **BULK_INSERT** command, set **hoodie.datasource.write.row.writer.enable** to **false** to enable Hoodie to use the Avro-compliant class for writing Parquet files.

13.15.6 Data in ro and rt Tables Cannot Be Synchronized to a MOR Table Recreated After Being Deleted Using Spark SQL

Question

After a MOR table is deleted using Spark SQL and then re-created, data in ro and rt tables cannot be synchronized to the MOR table in real time. The following error information is displayed:

```
WARN HiveSyncTool: Got runtime exception when hive syncing, but continuing as ignoreExceptions config is set
java.lang.IllegalArgumentException: Failed to get schema for table hudi_table2_ro does not exist
at org.apache.hudi.hive.HoodieHiveClient.getTableSchema(HoodieHiveClient.java:183)
at org.apache.hudi.hive.HiveSyncTool.syncHoodieTable(HiveSyncTool.java:286)
at org.apache.hudi.hive.HiveSyncTool.doSync(HiveSyncTool.java:213)
```

Answer

Cause:

To reduce access to Hive Metastore, a cache mechanism is added for Hudi tables. By default, data is cached for 1 hour. So, after a MOR table is deleted using Spark SQL and then recreated, data in ro and rt tables cannot be synchronized to the MOR table in real time.

Solution:

Set **hoodie.datasource.hive_sync.interval** to **0**.

```
set hoodie.datasource.hive_sync.interval=0;
```

13.15.7 IllegalArgumentException Is Reported When Kafka Is Used to Collect Data

Question

The error "org.apache.kafka.common.KafkaException: Failed to construct kafka consumer" is reported in the **main** thread, and the following error is reported.

```
java.lang.IllegalArgumentException: Could not find a 'KafkaClient' entry in the JAAS configuration. System property 'java.security.auth.login.config' is not set
```

Answer

This error may occur when you try to collect data from the Kafka source with SSL enabled and the installation program cannot read the **jars.conf** file and its properties.

To solve this problem, pass the required property as part of the command submitted through Spark. Example: **--files jaas.conf,failed_tables.json --conf 'spark.driver.extraJavaOptions=-Djava.security.auth.login.config=jaas.conf' --conf 'spark.executor .extraJavaOptions=-Djava.security.auth.login.config=jaas.conf'**

13.15.8 SQLException Is Reported During Hive Data Synchronization

Question

The following error is reported during Hive data synchronization:

```
Caused by: java.sql.SQLException: Error while processing statement: FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. Unable to alter table. The following columns have types incompatible with the existing columns in their respective positions :
__col1,__col2
```

Answer

This error usually occurs when you try to add a new column to an existing Hive table using the **HiveSyncTool.java** class. Databases usually do not allow the modification of a column data type from a higher order to lower order or cases where the data types may conflict with the data that is already stored or will be stored in the table. To solve this problem, set **hive.metastore.disallow.incompatible.col.type.changes** to **false**.

13.15.9 HoodieHiveSyncException Is Reported During Hive Data Synchronization

Question

The following error is reported during Hive data synchronization:

```
com.uber.hoodie.hive.HoodieHiveSyncException: Could not convert field Type from <type1> to <type2> for field col1
```

Answer

This error occurs because HiveSyncTool currently supports only few compatible data type conversions. The exception is thrown if any other incompatible changes are made.

Check the data type evolution for the related field and verify if it indeed can be considered as a valid data type conversion based on the Hudi code base.

13.15.10 SemanticException Is Reported During Hive Data Synchronization

Question

The following error is reported during Hive data synchronization:

```
org.apache.hadoop.hive.ql.parse.SemanticException: Database does not exist: test_db
```

Answer

This error typically occurs when Hive synchronization is performed on the Hudi data set but the configured **hive_sync** database does not exist.

Create the corresponding database on your Hive cluster and try again.

14 Using Hue

14.1 Accessing the Hue Web UI

Scenario

After Hue is installed in an MRS cluster, you can use Hadoop-related components on the Hue web UI.

This section describes how to open the Hue web UI on the MRS cluster.

NOTE

To access the Hue web UI, you are advised to use a browser that is compatible with the Hue WebUI, for example, Google Chrome 50. The Internet Explorer may be incompatible with the Hue web UI.

Impact on the System

Site trust must be added to the browser when you access Manager and Hue web UI for the first time. Otherwise, the Hue web UI cannot be accessed.

Prerequisites

When Kerberos authentication is enabled, you have been assigned the permission to access Hive by the MRS cluster administrator. For details, see **Creating a User**.For example, create a human-machine user **hueuser**, add it to the **hive** (primary group), **hadoop**, **supergroup** groups, and associate it with the **manager_view** role.

This user is used to log in to Manager.

Procedure

Step 1 Log in to the service page.

Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **Hue**.

Step 2 On the right of **Hue WebUI**, click the link to open the Hue web UI.

Hue WebUI provides the following functions:

- Click  to execute query statements of Hive and SparkSQL as well as Notebook code. Make sure that Hive and Spark2x have been installed in the MRS cluster before this operation.
- Click  to submit workflow tasks, scheduled tasks, and bundle tasks.
- Click  to view, import, export, and delete tasks on the Hue web UI, such as workflow tasks, scheduled tasks, and bundle tasks.
- Click  to manage metadata in Hive and SparkSQL. Make sure that Hive and Spark2x have been installed in the MRS cluster before this operation.
- Click  to view the directories and files in HDFS. Make sure that HDFS has been installed in the MRS cluster before this operation.
- Click  to view all jobs in the MRS cluster. Make sure that Yarn has been installed in the MRS cluster before this operation.
- Use  to create or query HBase tables. Make sure that the HBase component has been installed in the MRS cluster and the Thrift1Server instance has been added before this operation.
- Use  to import data that is in the CSV or TXT format.

NOTE

- When you log in to the Hue web UI as user **hueuser** for the first time, you need to change the password.
- After obtaining the URL for accessing the Hue web UI, you can give the URL to other users who cannot access MRS Manager for accessing the Hue web UI.
- If you perform operations on the Hue web UI only but not on Manager, you must enter the password of the current login user when you access Manager again.

----End

14.2 Creating a Hue Job

14.2.1 Using Hue to Execute HiveQL

Scenario

Hue provides the Hive management UI to run HiveQL (HQL) statements and query Hive data.

Access Editor

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 In the navigation tree on the left, click </> and choose **Hive**. The **Hive** page is displayed.

Hive supports the following functions:

- Running HQL statements

  Select the target database on the left. You can also click default in the upper right corner and enter the target database name to search for the database.

  Enter an HQL statement in the text box and click ▶ or press **Ctrl+Enter** to run the HQL statement. The execution result is displayed on the **Result** tab page.

- Analyzing HQL statements

  Select the target database on the left, enter the HQL statement in the text box, and click 🕮 to compile the HQL statement and check whether the statement is correct. The execution result is displayed under the text editing box.

- Saving HQL statements

  Enter the HQL statement in the text box, click 💾 in the upper right corner, and enter the name and description. You can view the saved statements in the **Saved Queries** tab.

- Viewing historical records

  Click **Query History** to view the HQL status. You can view the history of all the statements or only the saved statements. If many historical records exist, you can enter keywords in the text box to search for desired records.

- Configuring advanced queries

  Click ⚙ in the upper right corner to configure the file, function, and settings.

- Viewing the information of shortcut keys

  Click ? in the upper right corner to view information about all shortcut keys.

----End

How to Use Metadata Browser

Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

- Viewing metadata of Hive tables

  Click 🗄 in the navigation tree on the left and click a table name. The metadata of the Hive table is displayed.

- Managing metadata of Hive tables

  On the metadata information page of a Hive table:

  – Click **Import** in the upper right corner to import data.

  – Click **Overview** to view the location of the table file in the **Properties** field.

View the field information of each column in a Hive table and manually add description information. Note that the added description information is not the field comments in the Hive table.

  - Click **Sample** to browse data.

- Managing Hive metadata tables

  Click + in the left list to create a table based on the uploaded file in the database. You can also manually create a table.

CAUTION

The Hue page is used to view and analyze data such as files and tables. Do not perform high-risk management operations such as deleting objects on the page. If such an operation is required, perform the operation on each component after confirming that the operation does not affect services. For example, use the HDFS client to perform operations on HDFS files and use the Hive client to perform operations on Hive tables.

Executing HQL Statements

Step 1 Select a Hive database from the **Database** drop-down list box. The default database is **default**.

The system displays all available tables. You can enter a keyword of the table name to search for the desired table.

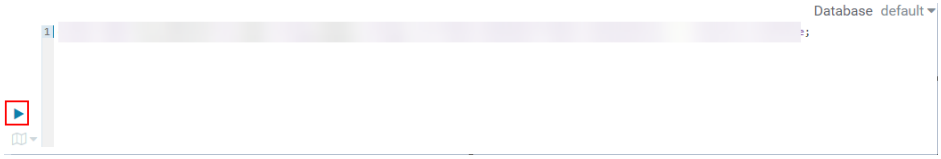
Step 2 Click the desired table name. All columns in the table are displayed.

Move the cursor to the row where the table or column is located and click . Column details are displayed.

Step 3 Enter the query statements in the area for editing HQL statements.

Step 4 Click to execute the HQL statements.

Figure 14-1 Executing a statement



NOTE

- If you want to use the entered HQL statements again, click to save them.
- Advanced query configuration:

  Click in the upper right corner to configure information such as files, functions, and settings.
- Viewing the information of shortcut keys:

  Click in the upper right corner to view the syntax and keyboard shortcut information.
- To delete an entered HQL statement, click the triangle next to and select **Clear**.
- Viewing history:

  Click **Query History** to view the HQL running status. You can view the history of all the statements or only the saved statements. If many historical records exist, you can enter keywords in the text box to search for desired records.

----End

Viewing Execution Results

Step 1 View the execution results below the execution area on **Hive**. The **Query History** tab page is displayed by default.

Step 2 Click a result to view the execution result of the executed statement.

NOTE

Hue does not support the display of a large amount of data. When a large number of SQL query results are loaded, the page may be frozen and some data may not be displayed. It is recommended that no more than 5,000 lines of query results be loaded.

----End

Managing Query Statements

Step 1 Click **Saved Queries**.

Step 2 Click a saved statement. The system automatically adds the statement to the editing area.

----End

Modifying the Session Configuration of the Hue Editor

Step 1 On the editor page, click .

Step 2 Click on the right of **Files**, and then click to select files.

You can click next to **Files** to add a file resource.

Step 3 In the **Functions** area, enter a user-defined name and the class name of the function.

You can click next to **Functions** to add a customized function.

Step 4 In the **Settings** area, enter the Hive parameter name in the **Key**, and value in **Value**. The current Hive session connects to Hive based on the customized configuration.

You can click to add a parameter.

----End

Typical Scenario

On the Hue page, create a Hive table as follows:

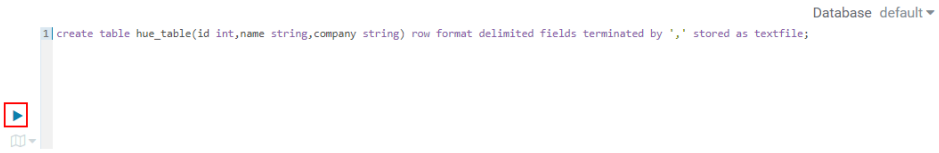
Step 1 Click at the upper left corner of Hue web UI and select the Hive instance to be operated to enter the Hive command execution page.

Step 2 Enter an HQL statement in the command input box, for example:

create table hue_table(id int,name string,company string) row format delimited fields terminated by ',' stored as textfile;

Click to execute the HQL statements.

Figure 14-2 Executing a statement

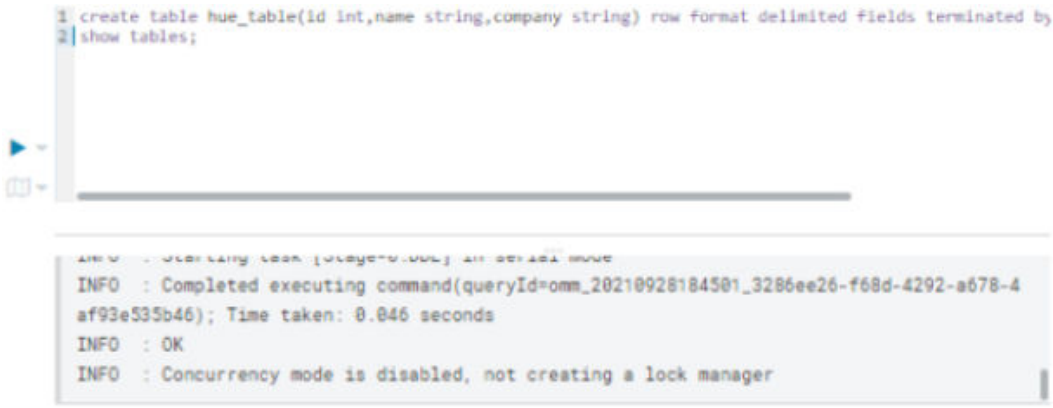


Step 3 Enter the following command in the command input box:

show tables;

Click to view the created table **hue_table** in **Result**.

Figure 14-3 Viewing the result



----End

14.2.2 Using Hue to Execute SparkSQL

Scenario

You can use Hue to execute SparkSQL statements in a cluster on the UI.

Configuring Spark2x

Before using the SparkSql editor, you need to modify the Spark2x configuration.

Step 1 Go to the Spark2x configuration page. For details, see **Modifying Cluster Service Configuration Parameters**.

Step 2 Set the Spark2x multi-instance mode. Search for and modify the following parameters of the Spark2x service:

Parameter	Value
spark.thriftserver.proxy.enabled	false
spark.scheduler.allocation.file	#{conf_dir}/fairscheduler.xml

Step 3 Go to the JDBCServer2x customization page and add the following customized items to the **spark.core-site.customized.configs** parameter:

Table 14-1 Custom parameters

Parameter	Value
hadoop.proxyuser.hue.groups	*
hadoop.proxyuser.hue.hosts	*

Step 4 Save the configuration and restart the meta and Spark2x services.

----End

Accessing the Editor

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 In the navigation tree on the left, click  and choose **SparkSql**. The **SparkSql** page is displayed.

SparkSql supports the following functions:

- Executes and manages SparkSql statements.
- Views the SparkSql statements saved by the current user in **Saved Queries**.
- Queries SparkSql statements executed by the current user in **Query History**.

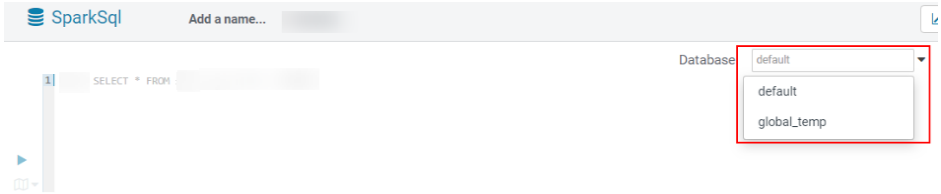
----End

Executing SparkSql Statements

Step 1 Select a SparkSql database from the **Database** drop-down list box. The default database is **default**.

The system displays all available tables. You can enter a keyword of the table name to search for the desired table.

Figure 14-4 Selecting a database



Step 2 Click the desired table name. All columns in the table are displayed.

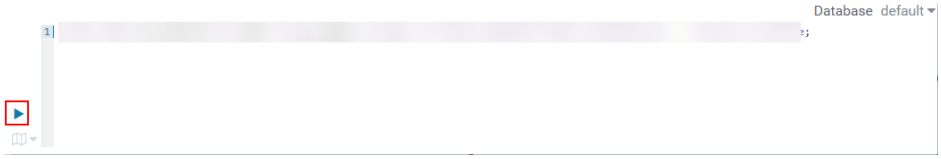
Move the cursor to the row of the table and click . Column details are displayed.

Step 3 In the SparkSql statement editing area, enter the query statement.

Click the triangle next to and select **Explain**. The editor checks the syntax and execution plan of the entered statements. If the statements have syntax errors, the editor reports **Error while compiling statement**.

Step 4 Click to execute the SparkSql statement.

Figure 14-5 Executing a statement



NOTE

- If you want to use the entered SparkSql statements again, click to save them.
- Advanced query configuration:

  Click in the upper right corner to configure information such as files, functions, and settings.
- Viewing the information of shortcut keys:

  Click in the upper right corner to view the syntax and keyboard shortcut information.
- To format the SparkSql statement, click the triangle next to and select **Format**.
- To delete an entered SparkSql statement, click the triangle next to and select **Clear**.
- Viewing historical records:

  Click **Query History** to view the SparkSql running status. You can view the history of all the statements or only the saved statements. If many historical records exist, you can enter keywords in the text box to search for desired records.

----End

Viewing Execution Results

Step 1 View the execution results below the execution area on **SparkSql**. The **Query History** tab page is displayed by default.

Step 2 Click a result to view the execution result of the executed statement.

----End

Managing Query Statements

Step 1 Click **Saved Queries**.

Step 2 Click a saved statement. The system automatically adds the statement to the editing area.

----End

14.2.3 Viewing Hive Metadata Using Hue

Scenario

You can use the Hue web UI to manage Hive metadata in an MRS cluster.

Using Metadata Manager

Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

- Viewing metadata of Hive tables

  Click in the navigation tree on the left and click a table name. The metadata of the Hive table is displayed.

- Managing metadata of Hive tables

  On the metadata information page of a Hive table:

  - Click **Import** in the upper right corner to import data.
  - Click **Overview** to view the location of the table file in the **PROPERTIES** field.

    View the field information of each column in a Hive table and manually add description information. Note that the added description information is not the field comments in the Hive table.
  - Click **Sample** to browse data.
- Managing Hive metadata tables

  Click  in the left list to create a table based on the uploaded file in the database. You can also manually create a table.

CAUTION

The Hue page is used to view and analyze data such as files and tables. Do not perform high-risk management operations such as deleting objects on the page. If such an operation is required, perform the operation on each component after confirming that the operation does not affect services. For example, use the HDFS client to perform operations on HDFS files and use the Hive client to perform operations on Hive tables.

14.2.4 Managing HDFS Files Using Hue

Scenario

Hue a UI-based file browser function for you to use the HDFS.

CAUTION

The Hue UI is used to view and analyze data such as files and tables. Do not perform high-risk management operations such as deleting objects. If such an operation is required, perform the operation on each component after confirming that the operation does not affect services. For example, use the HDFS client to perform operations on HDFS files and use the Hive client to perform operations on Hive tables.

Accessing File Browser

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 In the left navigation pane, click  . The **File Browser** page is displayed.

By default, the homepage of **File Browser** is the home directory of the current login user. On the displayed page, the following information about subdirectories for files in the directory is displayed:

Table 14-2 HDFS file attributes

Attribute	Description
Name	Name of a directory or file
Size	File size
User	Owner of a directory or file
Group	Group of a directory or file
Permission	Permission of a directory or file
Date	Time when a directory or file is created

Step 3 In the search box, enter a keyword. The system automatically searches directories or files in the current directory.

Step 4 Clear the search criteria. The system displays all directories or files.

----End

Creating a New File or Directory

Step 1 On the **File Browser** page, click **New**.

Step 2 Select an operation.

- **File**: creates a file. Enter a file name and click **Create**.
- **Directory**: creates a directory. Enter a directory name and click **Create**.

----End

Uploading User Files

Step 1 On the **File Browser** page, click **Upload**.

Step 2 In the displayed dialog box for uploading files, click **Select files** or drag the file to the dialog box.

----End

Managing Files or Directories

Step 1 On the **File Browser** page, select one or more directories or files.

Step 2 Click **Actions**. On the menu that is displayed, select an operation.

- **Rename**: renames a directory or file.
- **Move**: moves a file. In **Move to**, select a new directory and click **Move**.
- **Copy**: copies the selected files or directories.
- **Change permissions**: changes permission to access the selected directory or file.
  - You can grant the owner, the group, or other users with the **Read**, **Write**, and **Execute** permissions.

  - **Sticky**: indicates that only HDFS administrators, directory owners, and file owners can move files in the directory.
  - **Recursive**: indicates that permission is granted to subdirectories recursively.
- **Storage policies**: indicates the policies for storing files or directories in HDFS.
- **Summary**: indicates that the HDFS storage information about the selected file or directory can be viewed.

----End

Storage Policy Definition and Usage

NOTE

If the value of Hue parameter **fs_defaultFS** is set to **viewfs://ClusterX**, the big data storage policy cannot be enabled.

Storage policies on the Hue web UI are classified into the following two types:

- Static Storage Policies

  This is the currently used storage policy.

  According to the access frequency and importance of documents in HDFS, specify a storage policy for an HDFS directory, such as ONE_SSD or ALL_SSD. The files in this directory can be migrated to the storage media.
- Dynamic Storage Policies

  Set rules for an HDFS directory. The system can automatically change the storage policy, change the number of file copies, delete files, or move the file directory based on the latest access time and modification time of files. For details, see **Configuring HDFS Cold and Hot Data Migration**.

  Before configuring a dynamic storage policy on the Hue web UI, you must set the CRON expressions for cold and hot data migration and start automatic cold and hot data migration on Manager.

  Modify the following NameNode parameters of HDFS. For details, see **Modifying Cluster Service Configuration Parameters**.

Parameter	Description	Example Value
dfs.auto.data.mover.enable	Whether to enable automatic hot and cold data migration. The default value is **false**.	true
dfs.auto.data.mover.cron.expression	CRON expression for hot and cold data migration in HDFS, which is used to control the start time of data migration. This parameter is available only when **dfs.auto.data.mover.enable** is set to **true**. The default value is **0 * * * ***, indicating that the task is executed on the hour.	0 * * * *

Table 14-3 describes the expression for modifying the **dfs.auto.data.mover.cron.expression** parameter. * indicates consecutive time segments.

Table 14-3 Parameters in the execution expression

Column	Description
1	Minute. The value ranges from 0 to 59.
2	Hour. The value ranges from 0 to 23.
3	Date. The value ranges from 1 to 31.
4	Month. The value ranges from 1 to 12.
5	Week. The value ranges from 0 to 6. **0** indicates Sunday.

To set storage policies on the web UI, perform the following operations:

Step 1 Log in to FusionInsight Manager.

Step 2 On FusionInsight Manager, choose **System** > **Permission > Manage Role** > **Create Role**.

1. Set **Role Name**.
2. In the **Configure Resource Permission** area, choose *Name of the desired cluster* > **Hue**, select **Storage Policy Admin**, and click **OK**. Then, grant the permission to the role.

Step 3 Choose **System** > **Permission** > **User Group** > **Create User Group**. Set **Group Name** and click **Select and Add Role** next to **Role**. On the displayed page, select the role created in **Step 2** and click **OK** to add the role to the group.

Step 4 Choose **System** > **Permission** > **User** > **Create**.

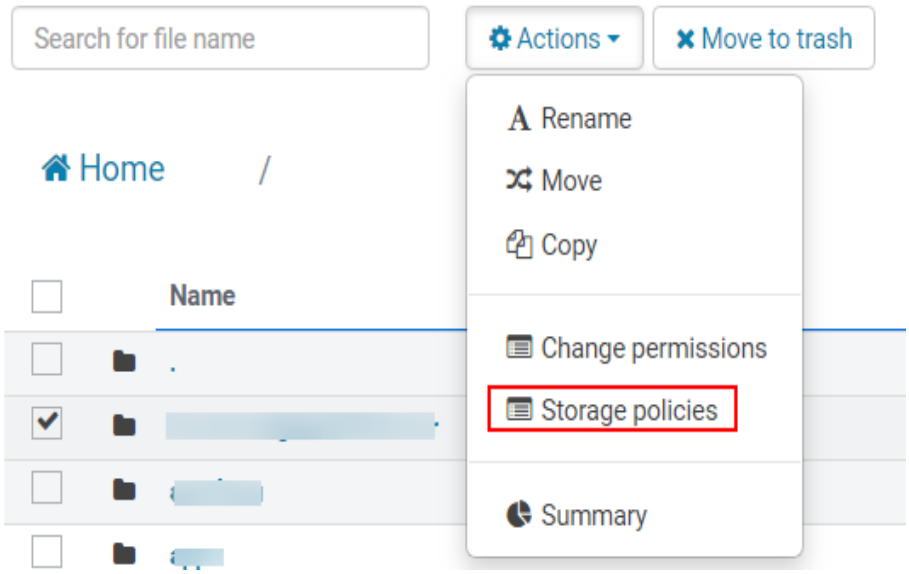
1. **Username**: Enter the name of the user to be added.
2. Set **User Type** to **Human-machine**.
3. Set **Password** and **Confirm Password** for logging in to the Hue web UI.
4. Click **Add** next to **User Group**. On the page that is displayed, select the user group created in **Step 3**, **supergroup**, **hadoop**, and **hive**, and click **OK**.
5. Set **Primary Group** to **hive**.
6. Click **Add** on the right of **Role**. On the page that is displayed, select the role created in **Step 2** and **System_administrator** role, and click **OK**.
7. Click **OK**. The user is added successfully.

Step 5 Access the Hue web UI as the created user. For details, see **Accessing the Hue Web UI**.

Step 6 In the left navigation tree, click  . The **File Browser** page is displayed.

Step 7 Select the check box of the directory and click **Actions** on the top of the page. Choose **Storage policies**.

Figure 14-6 Storage policies



Step 8 In the dialog box that is displayed, set a new storage policy and click **OK**.

- On the **Static Storage Policy** page, you can set a static storage policy and click **Save**.
- On the **Dynamic Storage Policy** page, you can create, delete, or modify a dynamic storage policy. **Table 14-4** describes the parameters.

Table 14-4 Parameters of the dynamic storage policy

Category	Parameter	Description
Rule	Last Access to File	Indicates the time when the file is last accessed.
	Last File Modification	Indicates the time when the file is last modified.
Operation	Change Number of Copies	Indicates the number of file copies.
	Modify Storage Policy	Indicates that you can modify storage policies to the following: HOT, WARM, COLD, ONE_SSD, and ALL_SSD.
	Move to Directory	Indicates that you can move the file to another directory.

NOTE

- You need to consider whether the rules conflict with each other and whether the rules damage the system when setting rules.
- When a directory is configured with multiple rules and operations, the rule that is triggered first is located at the bottom of the rule/operation list, and the rules that are triggered later are placed from bottom to top to prevent repeated operations.
- The system checks whether the files under the directory specified by the dynamic storage policy meet the rules on an hourly basis. If the files meet the rules, the execution is triggered. Execution logs are recorded in the **/var/log/Bigdata/hdfs/nn/hadoop.log** directory of the active NameNode.

----End

Typical Scenario

On the Hue page, view and edit HDFS files in text or binary mode as follows:

Viewing a File

Step 1 Access the Hue web UI.

Step 2 In the navigation pane on the left, click  . The **File Browser** page is displayed.

Step 3 Click the name of the file you want to view.

Step 4 Click **View as binary** to switch from the text mode to the binary mode. Click **View as file** to switch from the binary mode to the text mode.

Editing a file

Step 5 Click **Edit File**. The file content can be edited.

Step 6 Click **Save** or **Save As** to save the file.

----End

14.2.5 Managing Oozie Jobs Using Hue

Scenario

Users can use the Hue web UI to query all jobs in an MRS cluster.

Hue provides a UI for Oozie job management.

CAUTION

The Hue UI is used to view and analyze data such as files and tables. Do not perform high-risk management operations such as deleting objects. If such an operation is required, perform the operation on each component after confirming that the operation does not affect services. For example, use the HDFS client to perform operations on HDFS files and use the Hive client to perform operations on Hive tables.

Using Oozie Job Designer

Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

In the navigation pane on the left, click  and choose **Workflow**.

The job designer allows users to create MapReduce, Java, Streaming, Fs, SSH, Shell and DistCp jobs.

Using Editor

Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

In the navigation pane on the left, click  and choose **Workflow**.

Workflows, Schedule, and Bundle tasks can be created. Existing applications can be submitted for running, shared, copied, and exported.

- Each Workflow can contain one or more jobs to form a complete workflow for a specified service.

  When creating a Workflow, you can design jobs in the Hue editor and add the jobs to the Workflow.
- Each Schedule can define a time trigger to periodically execute a specified Workflow. One time trigger cannot execute multiple Workflows.
- Each Bundles can define a set to execute multiple Schedules so that different Workflows can be executed in batches.

Accessing Job Browser

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 Click  to enter the job browser.

View the jobs in the current cluster. View the status of the Workflow, Coordinator, and Bundles jobs.

NOTE

The number on **Job Browser** indicates the total number of jobs in the cluster.

Job Browser displays the following job information:

Table 14-5 MRS job attributes

Attribute	Description
Name	Job name
User	User who starts a job
Type	Job type
Status	Job status, including **Succeeded**, **Running**, and **Failed**.

Attribute	Description
Progress	Job running progress
Group	Group to which a job belongs
Start	Start time of a job
Duration	Job running duration
Id	Job ID, which is generated by the system automatically.

NOTE

If the MRS cluster has Spark, the **Spark-JDBCServer** job is started by default to execute tasks.

----End

Searching for Jobs

Step 1 In the search box of **Job Browser**, enter the specified character. The system automatically searches for all jobs that contain the keyword by ID, name, or user.

Step 2 Clear the search criteria. The system displays all jobs.

----End

Querying Job Details

Step 1 In the job list on the **Job Browser** page, click the row that contains the desired job to view details.

Step 2 On the **Metadata** tab page, you can view the metadata of the job.

NOTE

You can click **Log** to open the job running log.

----End

14.2.6 Managing HBase Tables Using Hue

Scenario

You can use Hue to create or query HBase tables in a cluster and run tasks on the Hue web UI.

NOTE

To operate HBase on the Hue web UI, the Thrift1Server instances of HBase must be deployed in the current MRS cluster.

The Thrift1Server instances are not installed by default. When you create a custom MRS cluster, you can select HBase and customize the cluster topology to add Thrift1Server instances. For details, see **Configuring Custom Topology**.

If the current cluster allows you to add services, you can deploy Thrift1Server instances when adding the HBase service for the first time. Then, restart the Hue service. For details, see **Managing Services**.

Accessing Job Browser

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 Click HBase . The **HBase Browser** page is displayed.

----End

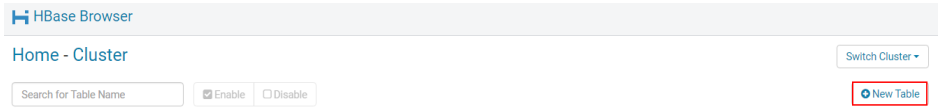
Creating an HBase Table

Step 1 Access the Hue web UI.

Step 2 Click HBase . The **HBase Browser** page is displayed.

Step 3 Click **New Table** on the right, enter the table name and column family parameters, and click **Submit**.

Figure 14-7 Creating a table



----End

Querying Data in an HBase Table

Step 1 Access the Hue web UI.

Step 2 Click HBase . The **HBase Browser** page is displayed.

Step 3 Click the HBase table to be queried. Then, click the key value next to search box in the upper part, and query the HBase table.

Figure 14-8 Searching by key value

----End

14.2.7 Using Hue to Execute HetuEngine SQL Statements

Scenario

You can use Hue to execute HetuEngine statements in a cluster on a UI.

Prerequisites

- The HetuEngine component has been installed in the MRS cluster and the HSFabric instance has been added. The Hue service needs to be restarted when HSFabric instances are added, deleted, migrated, or when ports are modified.
- A human-machine HetuEngine administrator user, for example, **hetu_user**, has been created in the cluster. For details, see **Creating a HetuEngine Permission Role**. For clusters with Ranger authentication enabled, the Ranger permission must be added to user **hetu_user** based on service requirements. For details, see **Adding a Ranger Access Permission Policy for HetuEngine**.
- A compute instance has been created and is running properly. For details, see **Creating a HetuEngine Compute Instance**.

Access the Editor

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 In the navigation pane on the left, click </> and choose **HetuEngine**. The **HetuEngine** page is displayed.

HetuEngine supports the following functions:

- Execute and manage HetuEngine SQL statements.
- View HetuEngine SQL statements saved by the current user in **Saved Queries**.
- Query HetuEngine SQL statements executed by the current user in **Query History**.

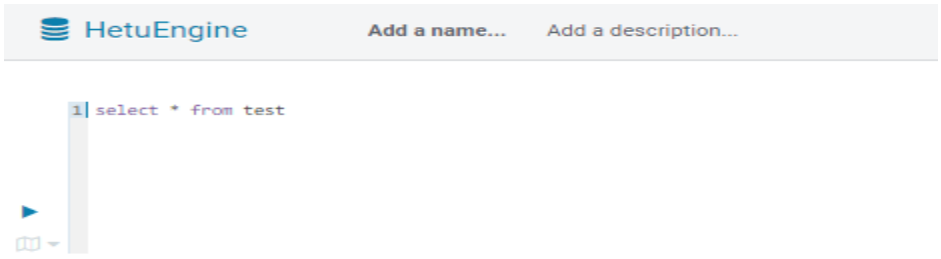
----End

Executing HetuEngine SQL Statements

Step 1 Enter a HetuEngine statement in the editor.

Step 2 Click ▶ to execute the HetuEngine SQL statement.

Figure 14-9 Executing a statement



NOTE

- Only one HetuEngine statement can be executed at a time on Hue.
- By default, **catalog=hive** and **schema=default** are used to execute a HetuEngine statement on Hue. You can use the SQL syntax **use** *<catalog>.<schema>* to switch the schema.
- If you want to use the entered HetuEngine SQL statement next time, click to save it.
- On the Hue page, you cannot specify a tenant to run a job. Instead, a default tenant is randomly selected from the tenant list associated with the user to run the job.
- Viewing history

  Click **Query History** to view the status of historical HetuEngine SQL statements. You can choose to view the status of all statements or only the saved statements. If there are many historical records, you can enter keywords in the text box to search for desired records.

----End

Viewing Execution Results

Step 1 View the execution results below the execution area on **HetuEngine**. The **Query History** tab is displayed by default.

Step 2 Click a result to view the execution result of the executed statement.

NOTE

Hue does not support the display of a large amount of data. When a large number of SQL query results are loaded, the page may be frozen and some data may not be displayed. It is recommended that no more than 5,000 lines of query results be loaded.

----End

14.3 Configuring HDFS Cold and Hot Data Migration

Scenario

The hot and cold data migration tool migrates HDFS files based on the configured policy. A policy is a set of conditional or non-conditional rules. If a file matches the rule set, the tool performs a group of operations for the file.

The hot and cold data migration tool supports the following rules and operations:

- Migration rules:
  - Data is migrated based on the latest access time of the file.
  - Data is migrated based on the file modification time.
  - Data is migrated without conditions.

Table 14-6 Rule condition tags

Condition Tag	Description
<age operator="lt">	Defines the conditions for changing the age or modification time.

Condition Tag	Description
<atime operator="gt">	Defines the condition for accessing time.

NOTE

For a manual migration rule, no condition is required.

- Operations:
  - Set the storage policy to a given data tier.
  - Migrate files to another folder.
  - Configure the number of copies for a file.
  - Delete a file.
  - Set a node label.

Table 14-7 Behavior types:

Behavior Type	Description	Required Parameters
MARK	Determines the data access frequency and set a data storage policy.	<param> <name>targettier</name> <value>STORAGE_POLICY</value> <param>
MOVE	Sets the data storage policy or NodeLabel and invokes the HDFS Mover tool.	<param> <name>targettier</name> <value>STORAGE_POLICY</value> <param> <param> <name>targetnodelabels</name> <value>SOME_EXPRESSION</value> <param> **NOTE** You can set either or both of the parameters.
SET_REPL	Configures the number of copies for a file.	<param> <name>replcount</name> <value>INTEGER</value> <param>

Behavior Type	Description	Required Parameters
MOVE_TO_FOLDER	Moves the file to the target folder. If **overwrite** is set to **true**, the target path will be overwritten.	<param> <name>target</name> <value>PATH</value> <param> <param> <name>overwrite</name> <value>true/false</value> <param> **NOTE** **overwrite** is an optional parameter. If this parameter is not set, the default value **false** is used.
DELETE	Delete a file.	N/A

Configuration Description

You must periodically invoke the migration tool and perform the following operations in the **hdfs-site.xml** file on the client:

Table 14-8 Parameter description

Parameter	Description	Default Value
dfs.auto-data-movement.policy.class	Specifies the default data migration policy. **NOTE** Currently, only **DefaultDataMovementPolicy** is supported.	com.xxx.hadoop.hdfs.datamovement.policy.DefaultDataMovementPolicy
dfs.auto.data.mover.id	Specifies the output file name of the hot and cold data migration policy.	Current system time (ms)
dfs.auto.data.mover.output.dir	Specifies the name of the HDFS directory to which cold and hot data is migrated. The migration tool writes the behavior status file here.	/system/datamovement

DefaultDataMovementPolicy has the configuration file **default-datamovement-policy.xml**. Users need to define all rules based on the age or access time and operations performed in this file. This file must be stored in **classpath** of the client.

The following is an example of the **default-datamovement-policy.xml** file:


```
<policies>
 <policy>
  <fileset>
   <file>
    <name>/opt/data/1.txt</name>
   </file>
   <file>
    <name>/opt/data/*/subpath/</name>
    <excludes>
     <name>/opt/data/some/subpath/sub1</name>
    </excludes>
   </file>
  </fileset>
  <rules>
   <rule>
    <age>2w</age>
    <action>
     <type>MOVE</type>
     <params>
      <param>
       <name>targettier</name>
       <value>HOT</value>
      </param>
     </params>
    </action>
   </rule>
  </rules>
 </policy>
</policies>
```

NOTE

Other attributes can be added to the tags used in policies, rules, and behavior operations. For example, **name** can be used to manage the mapping between the user UI (for example, Hue UI) and tool input XML.

Example: **<policy name="Manage_File1">**

The tags are described as follows:

Table 14-9 Description of configuring tags

Tag	Description	Reusable or Not
<policy>	Define a single policy. • **idempotent**: specifies whether to check the next rule if the current rule is met when multiple rules exist in the policy. Example: **<policy name ="policy2" idempotent ="true">** The default value is **true**, indicating that the rule and action are idempotent and you can continue to check the next rule. If the value is **false**, the evaluation stops at the current rule. • **hours_allowed**: indicates whether to execute policy evaluation based on the system time. The value of **hours_allowed** is a number separated by commas (,). The value ranges from 0 to 23, indicating the system time. Example: **<policy name ="policy1" hours_allowed ="2-6,13-14">** If the current system time is within the configured range, continue the evaluation. Otherwise, the evaluation will be skipped. **NOTE** In the input XML, only one policy is supported per file. Therefore, all rules in the file must be covered by a policy tag.	Yes
<fileset>	Define a group of files or folders for each policy.	No (in the **policy** tag)
<file>	One or more **<name>** tags are configured for the definition file and/or folder in the **<file>** tag. The file or folder name supports POSIX globs.	Yes (in the **fileset** tag)
<excludes>	Define this tag in the **<file>** tag. This tag can contain multiple **<name>** tags. In the file or folder range configured in the **<file>** tag, the files or folders contained in the **<name>** tag will be excluded. The file or folder name supports POSIX globs.	No (in the **fileset** tag)
<rules>	Specifies multiple rules defined for a policy.	No (in the **policy** tag)
<rule>	Specifies a single rule to be defined.	Yes (in the **rules** tag)

Tag	Description	Reusable or Not
<age>or<atime>	Defines the age/accesstime of the file defined in **<fileset>**. The policy matches the age. The value of **age** can be in the *[num]y[num]m[num]w[num]d[num]h* format. In the command, *num* indicates a number. The meanings of the letters are as follows: * *y*: year (365 days in a year) * *m*: month (30 days in a month) * *w*: week (7 days in a week) * *d*: day * *h*: hour You can use the year, month, week, day, or hour independently, or you can combine them. For example, **1y2d** indicates one year and two days or 367 days. If there is no unit (that is, the number is not followed by any letter), the default unit is day. **NOTE** You can configure **gt** (greater) and **lt** (less) in the **<age>** and **<atime>** tags. The default operator is **gt**. Example: **<age operator="lt">**	No (in the **rule** tag)
<action>	If the rule is matched, this tag defines the action to be executed.	No (in the **rule** tag)
<type>	Defines the action type. Currently, the supported action types are MOVE and MARK.	No (in the **action** tag)
<**params**>	Defines parameters related to each action.	No (in the **action** tag)
<param>	Defines a name-value format parameter that uses the **<name>** and **<value>** tags. For MARK and MOVE, only the **targettier** parameter is supported. This parameter specifies the data storage policy if the age rule is met. If multiple parameters have the same name, the first parameter value is used. For marks, the supported **targettier** values are **ALL_SSD**, **ONE_SSD**, **HOT**, **WARM**, and **COLD**. For MOVE, the supported **targettier** values are **ALL_SSD**, **ONE_SSD**, **HOT**, **WARM**, and **COLD**.	Yes (in the **params** tag)

For files or folders under the **<file>** tag, the **FileSystem#globStatus** API is used. For other files or folders, the **GlobPattern** class (used by GlobFilter) is used. For details, see the description of supported APIs. For example, for globStatus, **/opt/**

hadoop/* will match everything in the **/opt/hadoop** folder. **/opt/*/hadoop** matches all hadoop folders in the subdirectories of the**/opt** directory.

For globStatus, the glob mode of each path component is matched. For other components, the glob mode is directly matched.

Versions earlier than MRS 3.2.0: **https://hadoop.apache.org/docs/r3.1.1/api/org/apache/hadoop/fs/FileSystem.html#globStatus(org.apache.hadoop.fs.Path)**

MRS 3.2.0 or later: **https://hadoop.apache.org/docs/r3.3.1/api/org/apache/hadoop/fs/FileSystem.html#globStatus(org.apache.hadoop.fs.Path)**

Glob	Name	Matches
*	*asterisk*	Matches zero or more characters
?	*question mark*	Matches a single character
[ab]	*character class*	Matches a single character in the set {a, b}
[^ab]	*negated character class*	Matches a single character that is not in the set {a, b}
[a-b]	*character range*	Matches a single character in the (closed) range [a, b], where a is lexicographically less than or equal to b
[^a-b]	*negated character range*	Matches a single character that is not in the (closed) range [a, b], where a is lexicographically less than or equal to b
{a,b}	*alternation*	Matches either expression a or b
\c	*escaped character*	Matches character c when it is a metacharacter

Behavior Operation Example

- MARK

```
<action>
 <type>MARK</type>
 <params>
  <param>
   <name>targettier</name>
   <value>HOT</value>
  </param>
 </params>
</action>
```

- MOVE

```
<action>
 <type>MOVE</type>
 <params>
  <param>
   <name>targettier</name>
   <value>HOT</value>
  </param>
  <param>
   <name>targetnodelabels</name>
   <value>SOME_EXPRESSION</value>
  </param>
 </params>
</action>
```

- SET_REPL

```
<action>
 <type>SET_REPL</type>
 <params>
  <param>
   <name>replcount</name>
   <value>5</value>
  </param>
```



```
  </params>
</action>
```

- MOVE_TO_FOLDER

```
<action>
  <type>MOVE_TO_FOLDER</type>
  <params>
    <param>
      <name>target</name>
      <value>path</value>
    </param>
    <param>
      <name>overwrite</name>
      <value>true</value>
    </param>

  </params>
</action>
```

NOTE

The **MOVE_TO_FOLDER** operation only changes the file path to the target folder and does not change the block location. If you want to move a block, you need to configure an independent move policy.

- DELETE

```
<action>
  <type>DELETE</type>
</action>
```

NOTE

- When writing an XML file, pay attention to the configuration and sequence of behavior operations. The hot and cold data migration tool executes the rules in the sequence specified in the input XML file.
- If you want to run only one rule based on **atime/age**, sort the rules in descending order of time and set the idempotent attribute to false.
- If the delete operation is configured for a file set, other rules cannot be configured after the delete operation is performed.
- The **-fs** option can be used to specify the default file system address of the client.

Audit Logs

The cold and hot data migration tool supports audit logs of the following operations:

- Tool startup status
- Behavior type, parameter details, and status
- Tool completion status

To enable the audit log tool, add the following attributes to the **<HADOOP_CONF_DIR>/log4j.property** file:

```
autodatatool.logger=INFO, ADMTRFA
autodatatool.log.file=HDFSAutoDataMovementTool.audit
log4j.logger.com.xxx.hadoop.hdfs.datamovement.HDFSAutoDataMovementTool.audit=${autodatatool.logger}
log4j.additivity.com.xxx.hadoop.hdfs.datamovement.HDFSAutoDataMovementTool-audit=false
log4j.appender.ADMTRFA=org.apache.log4j.RollingFileAppender
log4j.appender.ADMTRFA.File=${hadoop.log.dir}/${autodatatool.log.file}
log4j.appender.ADMTRFA.layout=org.apache.log4j.PatternLayout
log4j.appender.ADMTRFA.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
log4j.appender.ADMTRFA.MaxBackupIndex=10
log4j.appender.ADMTRFA.MaxFileSize=64MB
```


NOTE

For details, see the **<HADOOP_CONF_DIR>/log4j_autodata_movment_template.properties** file.

14.4 Typical Hue Parameters

Page Access

Go to the **All Configurations** page of the Hue service by referring to **Modifying Cluster Service Configuration Parameters**.

Parameter Description

For details about Hue common parameters, see **Table 14-10**.

Table 14-10 Hue common parameters

Configuration	Description	Default Value	Value Range
HANDLER_ACCESSLOG_LEVEL	Hue access log level.	DEBUG	• ERROR • WARN • INFO • DEBUG
HANDLER_AUDITSLOG_LEVEL	Hue audit log level.	DEBUG	• ERROR • WARN • INFO • DEBUG
HANDLER_ERRORLOG_LEVEL	Hue error log level.	ERROR	• ERROR • WARN • INFO • DEBUG
HANDLER_LOGFILE_LEVEL	Hue run log level.	INFO	• ERROR • WARN • INFO • DEBUG
HANDLER_LOGFILE_MAXBACKUPINDEX	Maximum number of Hue log files.	20	1 to 999
HANDLER_LOGFILE_SIZE	Maximum size of a Hue log file.	5 MB	-

For details about Hue custom parameters, see **Table 14-11**. The following custom parameters are available only for MRS 3.1.2 or later.

Table 14-11 Hue custom parameters

Configuration	Description
dfs.customized.configs	Custom configuration item added to the global configuration file **hdfs-site.xml**.
hbase.customized.configs	Custom configuration item added to the global configuration file **hbase-site.xml**.
hive.customized.configs	Custom configuration item added to the global configuration file **hive-site.xml**.

14.5 Hue Log Overview

Log Description

Log paths: The default paths of Hue logs are **/var/log/Bigdata/hue** (for storing run logs) and **/var/log/Bigdata/audit/hue** (for storing audit logs).

Log archive rules: The automatic compression and archiving function of the Hue logs is enabled. By default, when the size of a log file (**access.log**, **error.log**, **runcpserver.log**, or **hue-audits.log**) exceeds 5 MB, logs are automatically compressed. A maximum of 20 latest compressed files are reserved. The number of compressed files and compression threshold can be configured.

Table 14-12 Hue log list

Type	Log File Name	Description
Run log	access.log	Access log file
	error.log	Error log file
	gsdb_check.log	Log file of the GaussDB check information
	kt_renewer.log	Log file of Kerberos authentication
	kt_renewer.out.log	Log file of the abnormal Kerberos authentication logs
	runcpserver.log	Log file of operation records
	runcpserver.out.log	Log file of process running exceptions

Type	Log File Name	Description
	supervisor.log	Log file of process startup
	supervisor.out.log	Log file of process startup exceptions
	dbDetail.log	Log file of database initialization
	initSecurityDetail.log	Download initialization log file of the Keytab file
	postinstallDetail.log	Work log file generated after the Hue service is installed
	prestartDetail.log	Prestart log file
	statusDetail.log	Log file of the Hue health status
	startDetail.log	Startup log
	get-hue-ha.log	Log file of the Hue HA status
	hue-ha-status.log	Log file of the Hue HA status monitoring
	get-hue-health.log	Log file of the Hue health status
	hue-health-check.log	Log file of the Hue health check
	hue-refresh-config.log	Log file of the Hue configuration update
	hue-script-log.log	Log file of the Hue operations on the Manager console
	hue-service-check.log	Log file of the Hue service status monitoring
	db_pwd.log	Log that records the changes of the password for Hue to connect to the DBService database
	modifyDBPwd_*Date*.log	-
	watch_config_update.log	Parameter update log file
Audit log	hue-audits.log	Audit log file

Log Level

Table 14-13 describes the log levels supported by Hue.

Levels of logs are ERROR, WARN, INFO, and DEBUG from the highest to the lowest priority. Run logs of equal or higher levels are recorded. The higher the specified log level, the fewer the logs recorded.

Table 14-13 Log levels

Level	Description
ERROR	Logs of this level record error information about system running.
WARN	Logs of this level record exception information about the current event processing.
INFO	Logs of this level record normal running status information about the system and events.
DEBUG	Logs of this level record the system information and system debugging information.

To modify log levels, perform the following operations:

Step 1 Go to the **All Configurations** page of the Hue service by referring to **Modifying Cluster Service Configuration Parameters**.

Step 2 In the navigation tree on the left, select **Log** corresponding to the role to be modified.

Step 3 Select the log level to be changed on the right.

Step 4 Save the configuration. In the displayed dialog box, click **OK** to make the configurations take effect.

Step 5 Restart the service or instance whose configuration has expired for the configuration to take effect.

----End

Log Format

The following table lists the Hue log formats:

Table 14-14 Log formats

Type	Format	Example
Run log	*<dd-MM-yy HH:mm:ss,SSS><Location where the log event occurs><Log level><Message in the log>*	[03/Nov/2014 11:57:19] middleware | INFO | Unloading MimeTypeJSFileFixStrea-mingMiddleware.

Type	Format	Example
	<Log level><Time format><yyyy-MM-dd HH:mm:ss,SSS><Location where the log event occurs><Message in the log>	INFO : CST 2014-11-06 11:22:52 hue-ha-status.sh : update 4 <= 15:myHostName=10.0.0.250 ACTIVE=10.0.0.250
Audit log	*<UserName><yyyy-MM-dd HH:mm:ss,SSS>< Audit operation description> <Resource parameter> <URL> <Whether to allow> <Audit operation> <IP address>*	{"username": "admin", "eventTime": "2014-11-06 10:28:34", "operationText": "Successful login for user: admin", "service": "accounts", "url": "/accounts/login/", "allowed": true, "operation": "USER_LOGIN", "ipAddress": "10.0.0.250"}

14.6 Common Issues About Hue

14.6.1 Why Do HQL Statements Fail to Execute in Hue Using Internet Explorer?

Question

What do I do if all HQL statements fail to be executed when I use Internet Explorer to access Hive Editor in Hue and following error message is displayed?

```
There was an error with your query.
```

Answer

Internet Explorer does not support processing of AJAX POST requests containing form data in 307 redirection.

You are advised to use a compatible browser, for example, Google Chrome.

14.6.2 How Do I Solve the Problem of Setting the Time Zone of the Oozie Editor on the Hue Web UI?

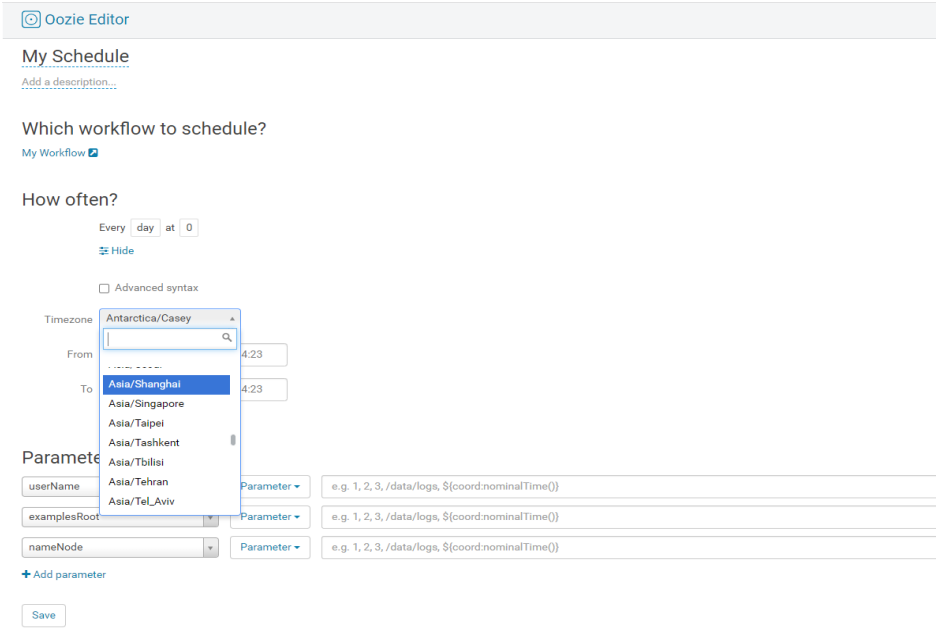
Question

How do I solve the problem that some time zone settings cause task submission failure when setting the Oozie time zone on Hue?

Answer

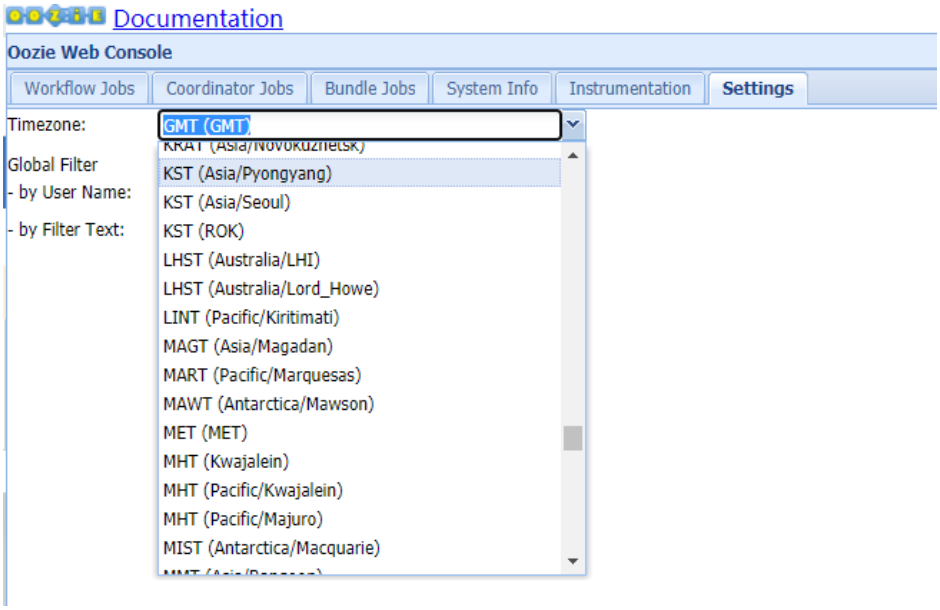
Adaptation issues exist in some time zones. You are advised to select **Asia/Shanghai**, as shown in **Figure 14-10**.

Figure 14-10 Selecting a time zone



For details about the supported time zones, see **Timezone** on the **Settings** tab page of the Oozie web UI, as shown in **Figure 14-11**.

Figure 14-11 Supported time zones



14.7 Hue Troubleshooting

14.7.1 Why Does the use database Statement Become Invalid in Hive?

Question

When Hive is used, the **use database** statement is entered in the text box to switch the database, and other statements are also entered, why does the database fail to be switched?

Answer

Using Hive on Hue is different from using Hive on the Hive client. There is an option to select a database on the Hue interface, and the database where the current SQL is executed is the one that is displayed on the interface.

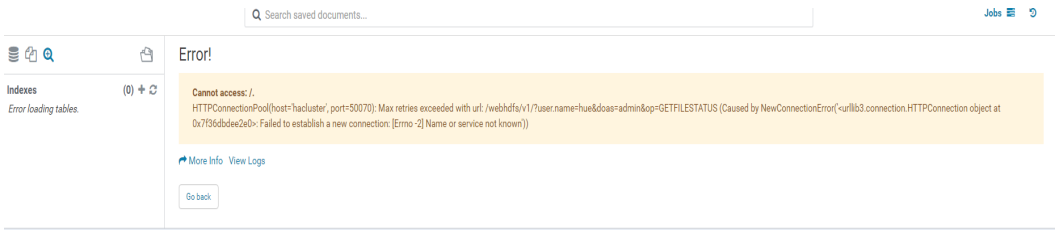
You are advised to use functions on the Hue interface instead of using statements to perform session-level and one-off operations, for example, setting parameters.

If you must enter specific statements to perform an operation, ensure that all statements you enter are in one text box.

14.7.2 Why Do HDFS Files Fail to Access Through the Hue Web UI?

Question

What can I do if an error message shown in the following figure is displayed, indicating that the HDFS file cannot be accessed when I use Hue web UI to access the HDFS file?



Answer

1. Check whether the user who logs in to the Hue web UI has the permissions of the **hadoop** user group.
2. Check whether the HttpFS instance has been installed for the HDFS service and is running properly. If the HttpFS instance is not installed, manually install and restart the Hue service.

14.7.3 Why Do Large Files Fail to Upload on the Hue Page

Question

What can I do when a large file fails to be uploaded on the Hue page?

Answer

1. You are advised to run commands on the client to upload large files instead of using the Hue file browser.
2. If you must use Hue to upload the file, perform the following steps to modify Httpd parameters:
   a. Log in to the active management node as user **omm**.
   b. Run the following command to edit the **httpd.conf** file:

      vi $BIGDATA_HOME/om-server/Apache-httpd-*/conf/httpd.conf
   c. Search for **21201** and add **RequestReadTimeout handshake=0 header=0 body=0** to the **</VirtualHost>** configuration, as shown in the following:

```
...
<VirtualHost *:21201>
    ServerName https://10.112.16.93:21201
    AllowEncodedSlashes On
    SSLProxyEngine On
    ProxyRequests Off
    TraceEnable off
    ProxyTimeout  1200
    RewriteEngine on
    RewriteMap proxylist dbm:${BIGDATA_ROOT_HOME}/om-server_*/Apache-httpd-*/conf/
proxylist.dbm

    RewriteRule ^(\/.*)$  ${proxylist:/Hue/Hue/21201}$1 [E=TARGET_PATH:$1,L,P]

    Header edit Location ^(?!https://10.112.16.93:20009|https://
10.112.16.93:21201)http[s]?://[^/]*(.*)$  https://10.112.16.93:21201$1

    ProxyPassReverseCookiePath / / interpolate

    SSLEngine On
    SSLProxyProtocol  All +TLSv1.2 -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
    SSLProtocol ALL +TLSv1.2 -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
    SSLCipherSuite ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:DHE-DSS-
AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:DHE-DSS-AES128-GCM-SHA256:DHE-
RSA-AES128-GCM-SHA256
    SSLProxyCheckPeerName off
    SSLProxyCheckPeerCN off
    SSLCertificateFile "${BIGDATA_ROOT_HOME}/om-server_*/Apache-httpd-*/conf/security/
proxy_ssl.cert"
    SSLCertificateKeyFile "${BIGDATA_ROOT_HOME}/om-server_*/Apache-httpd-*/conf/security/
server.key"
    SSLProxyCACertificateFile ${BIGDATA_ROOT_HOME}/om-server_*/apache-tomcat-*/conf/
security/tomcat.crt
    SSLCertificateChainFile "${BIGDATA_ROOT_HOME}/om-server_*/Apache-httpd-2.4.39/conf/
security/proxy_chain.cert"
    RequestReadTimeout handshake=0 header=0 body=0
</VirtualHost>
...
```

   d. Run the **pkill -9 httpd** command to stop the httpd process and wait for it to automatically restart.

14.7.4 Why Is the Hue Native Page Cannot Be Properly Displayed If the Hive Service Is Not Installed in a Cluster?

Question

Why is the native Hue page blank if the Hive service is not installed in a cluster?

Answer

In MRS 3.x and later versions, Hue depends on the Hive component. If the Hive service is not installed in a cluster, the native Hue page is blank.

If this problem occurs, check whether the Hive component is installed in the current cluster. If not, install the Hive component.

14.7.5 What Should I Do If It Takes a Long Time to Access the Native Hue UI and the File Browser Reports "Read timed out"?

Symptom

What should I do if it takes a long time to access the native Hue UI and the file browser reports "Read timed out"?

Answer

Check whether the HttpFS instance is installed in the HDFS service.

- If no, contact O&M support.
- If yes, restart the HttpFS instance.

15 Using Impala

15.1 Using the Impala Client

Impala is a massively parallel processing (MPP) SQL query engine for processing vast amounts of data stored in Hadoop clusters. It is an open source software written in C++ and Java. It provides high performance and low latency compared with other SQL engines for Hadoop.

Background

Suppose a user develops an application to manage users who use service A in an enterprise. The procedure of operating service A on the Impala client is as follows:

Operations on common tables:

- Create the **user_info** table.
- Add users' educational backgrounds and titles to the table.
- Query user names and addresses by user ID.
- Delete the user information table after service A ends.

Table 15-1 User information

No.	Name	Gender	Age	Address
12005000201	A	Male	19	City A
12005000202	B	Female	23	City B
12005000203	C	Male	26	City C
12005000204	D	Male	18	City D
12005000205	E	Female	21	City E
12005000206	F	Male	32	City F
12005000207	G	Female	29	City G

No.	Name	Gender	Age	Address
12005000208	H	Female	30	City H
12005000209	I	Male	26	City I
12005000210	J	Female	25	City J

Prerequisites

- The client has been installed. For example, the client is installed in the **/opt/hadoopclient** directory. The client directory in the following operations is only an example. Change it to the actual installation directory.
- For MRS 3.x and later, install Python2 on the Impala client node (running EulerOS 2.9 or later). For details, see **Installing Python2 on the Impala Client** .

Procedure

Step 1 Log in to the node where the client is installed as the client installation user.

Step 2 Run the following command to go to the client installation directory:

cd /opt/hadoopclient

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Run the Impala client command to implement service A.

- **Operations on internal tables**:

  Run the Impala client command **impala-shell**.

  NOTE

  By default, **impala-shell** attempts to connect to the Impala daemon on port 21000 of **localhost**. To connect to another host, use the **-i < host:port >** option, for example, **impala-shell -i** *xxx.xxx.xxx.xxx***:21000**. To automatically connect to a specific Impala database, use the **-d <database>** option. For example, if all your Kudu tables are in the **impala_kudu** database, **-d impala_kudu** can use this database. To exit Impala Shell, run the **quit** command.

  a. Create the **user_info** user information table according to **Table 15-1** and add data to it.

  ```
  create table user_info(id string,name string,gender string,age int,addr string);
  insert into table user_info(id,name,gender,age,addr) values("12005000201", "A", "Male", 19, "City A");
  ```

  ... (Other statements are the same.)

  b. Add users' educational backgrounds and titles to the **user_info** table.

  For example, to add educational background and title information about user 12005000201, run the following commands.

  ```
  alter table user_info add columns(education string,technical string);
  ```

  c. Query user names and addresses by user ID.

For example, to query the name and address of user 12005000201, run the following command:

```
select name,addr from user_info where id='12005000201';
```

d. Delete the user information table:

```
drop table user_info;
```

- **Operations on external partition tables**:

  Create an external partition table and import data.

  a. Create a path for storing external table data.

  kinit *hive* (Run this command only in security mode.)

  NOTE

  The user must have the hive administrator permissions.

  hdfs dfs -mkdir /hive

  hdfs dfs -mkdir /hive/user_info

  b. Create a table.

  impala-shell

  NOTE

  By default, **impala-shell** attempts to connect to the Impala daemon on port 21000 of **localhost**. To connect to another host, use the **-i < host:port >** option, for example, **impala-shell -i** *xxx.xxx.xxx.xxx***:21000**. To automatically connect to a specific Impala database, use the **-d <database>** option. For example, if all your Kudu tables are in the **impala_kudu** database, **-d impala_kudu** can use this database. To exit Impala Shell, run the **quit** command.

```
create external table user_info(id string,name string,gender string,age int,addr string)
partitioned by(year string) row format delimited fields terminated by ' ' lines terminated by '\n'
stored as textfile location '/hive/user_info';
```

  NOTE

  - **fields terminated** indicates delimiters, for example, spaces.
  - **lines terminated** indicates line breaks, for example, **\n**.
  - **/hive/user_info** indicates the path of the data file.

  c. Import data.

  i. Execute the **insert** statement to insert data.

```
insert into user_info partition(year="2018") values ("12005000201", "A", "Male", 19, "City A");
```

  ii. Run the **load data** command to import file data.

  1) Create a file based on the data in **Table 15-1**. For example, the file name is **txt.log**. Fields are separated by space, and the line feed characters are used as the line breaks.

  2) Upload the file to HDFS.

  hdfs dfs -put txt.log /tmp

  3) Load data to the table.

  load data inpath '/tmp/txt.log' into table user_info partition (year='2018');

d. Query the imported data:
```
select * from user_info;
```

e. Delete the user information table:
```
drop table user_info;
```

----End

15.2 Accessing the Impala Web UI

You can view Impala job information on the Impala web UI. Impala web UIs are classified into the following types based on instances:

- **StateStore WebUI**: used to manage nodes.
- **Catalog WebUI**: used to view metadata.

Prerequisites

Impala has been installed in a cluster.

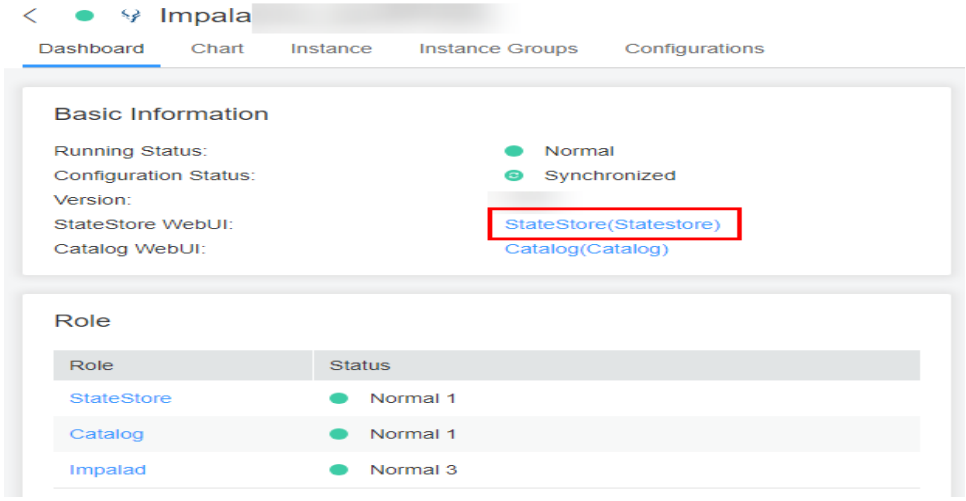
Accessing the StateStore Web UI

Step 1 Access Manager. For details, see **Accessing FusionInsight Manager**.

Step 2 Choose **Services** > **Impala**.

Step 3 On the **Dashboard** page, click **StateStore(Statestore)** next to **StateStore WebUI** in the **Basic Information** area to open the StateStore web UI.

Figure 15-1 StateStore WebUI



----End

Accessing the Catalog Web UI

Step 1 Access Manager. For details, see **Accessing FusionInsight Manager**.

Step 2 Choose **Services** > **Impala**.

Step 3 On the **Dashboard** page, click **Catalog(Catalog)** next to **Catalog WebUI** to open the Catalog web UI.

----End

15.3 Using Impala to Operate Kudu Tables

You can use the SQL statements of Impala to insert, query, update, and delete data in Kudu as an alternative to using Kudu APIs to build custom Kudu applications.

Prerequisite

A complete cluster client has been installed. For example, the installation directory is **/opt/Bigdata/client**. The client directory in the following operations is only an example. Replace it with the actual installation directory.

Impala on Kudu

Step 1 Log in to the node where the client is installed.

Step 2 Run the following command to initialize environment variables:

source /opt/Bigdata/client/bigdata_env

Step 3 If Kerberos authentication is enabled for the cluster, perform the following operation to authenticate the user. If Kerberos authentication is not enabled for the cluster, skip this step.

kinit *Service user*

Step 4 Run the following command to log in to the Impala client:

impala-shell

NOTE

By default, **impala-shell** attempts to connect to the Impala daemon on port 21000 of **localhost**. To connect to another host, use the **-i <host:port>** option. To automatically connect to a specific Impala database, use the **-d <database>** option. For example, if all your Kudu tables are in the **impala_kudu** database, **-d impala_kudu** can use this database. To exit the Impala shell, run the **quit** command.

Step 5 Run the following commands to create an Impala table and import the prepared data, for example, data in the **/tmp/data10** directory:

create table dataorigin (name string,age string,pt string, date_p date) row format delimited fields terminated by ',' stored as textfile;

load data inpath '/tmp/data10' overwrite into table dataorigin;

Step 6 Run the following command to create a Kudu table. In the command, **kudu.master_addresses** indicates the IP address of the KuduMaster instance. Set it to the actual IP address.

create table dataorigin2 (name string,age string,pt string, date_p date, primary key(name)) stored as kudu

TBLPROPERTIES('kudu.master_addresses'='192.168.190.164:7051,192.168.204.178:7051,192.168.244.63:7051');

Step 7 Perform the following operations on the Kudu table.

1. Insert data.

   insert into dataorigin2 select * from dataorigin;

2. Update data.

   UPDATE dataorigin2 SET date_p="2021-03-31" where age="73";

3. Upsert rows.

   UPSERT INTO dataorigin2 VALUES ("spjted","75","28","2021-03-32");

   UPSERT INTO dataorigin2 VALUES ("kwhakb","92","29","2021-03-33");

   UPSERT INTO dataorigin2 VALUES ("oftrkf","13","30","2021-03-34");

   UPSERT INTO dataorigin2 VALUES ("kiewti","36","31","2021-03-35");

   UPSERT INTO dataorigin2 VALUES ("rknmql","98","32","2021-03-36");

   UPSERT INTO dataorigin2 VALUES ("fwcoij","52","33","2021-03-37");

   UPSERT INTO dataorigin2 VALUES ("pgvpdo","37","34","2021-03-35");

4. Delete a row.

   DELETE FROM dataorigin2 WHERE date_p="2021-03-31";

----End

15.4 Interconnecting Impala with External LDAP

This section applies to MRS 3.1.0 or later.

Step 1 Log in to Manager.

Step 2 On Manager, choose **Cluster** > *Name of the desired cluster* > **Services** > **Impala** > **Configurations** > **All Configurations** > **Impalad(Role)** > **LDAP**.

Step 3 Set the following parameters.

Table 15-2 Parameter configuration

Parameter	Description	Remarks
--enable_ldap_auth	Whether to enable LDAP authentication	Value: **true** or **false**
--ldap_bind_pattern	LDAP user DN pattern	Example: **cn=#UID,ou=People,dc=huawei,dc=com** or **cn=%s,ou=People,dc=huawei,dc=com**

Parameter	Description	Remarks
--ldap_passwords_in_clear_ok	Whether the LDPA password is sent in plaintext	If this parameter is set to **true**, the LDAP password can be sent in plaintext. Value: **true** or **false** **NOTE** If **--enable_ldap_auth** is set to **true**, the LDAP TLS protocol is disabled by default during authentication. Therefore, you need to set **--ldap_passwords_in_clear_ok** to **true**. Otherwise, the Impalad role will fail to be started. To enable the Ldap TLS protocol, set **--ldap_tls** to **true** in the customized configuration of the Impalad role. After the configuration, the password can be sent in ciphertext.
--ldap_uri-ip	LDAP IP address	-
--ldap_uri-port	LDAP port number	Default value: **389**

Step 4 After the modification, click **Save** in the upper left corner. In the displayed dialog box, click **OK**.

Step 5 Choose **Cluster** > *Name of the desired cluster* > **Services** > **Impala** > **Instance**. On the displayed page, select the instances whose **Configuration Status** is **Expired**, choose **More** > **Restart Instance**, and restart the instance.

----End

15.5 Enabling and Configuring a Dynamic Resource Pool for Impala

This section describes how to enable and configure a dynamic resource pool to control Impala concurrency.

Background

Use a dynamic resource pool to control Impala concurrency.

Pool Config

Property	Value
Max memory (cluster wide)	1048576
Max concurrent queries	-1
Max queue size	200
Queue Timeout (ms)	60000
Min Query MEM_LIMIT range	0
Max Query MEM_LIMIT range	0
Clamp MEM_LIMIT query option	true

1. Log in to the master1 node of the cluster, switch to user **omm**, and create the **fair-scheduler.xml** and **llama-site.xml** files in the **/home/omm** directory.

```
[omm@node-master1IoKo impala]$ ll
total 16
-rw-------. 1 omm wheel  708 May 11 23:40 fair-scheduler.xml
-rw-------. 1 omm wheel 1062 May 11 23:53 llama-site.xml
-rw-------. 1 omm wheel 1118 May 11 23:12 llama-site.xml.bak
-rw-------. 1 omm wheel  572 May 11 23:32 update_config.sh
[omm@node-master1IoKo impala]$
```

2. Open the **fair-scheduler.xml** file and add the following configurations:

```
<allocations>
   <queue name="root">
      <aclSubmitApps> </aclSubmitApps>
      <queue name="default">
         <maxResources>4096 mb, 0 vcores</maxResources><!--For reference only-->
         <aclSubmitApps>*</aclSubmitApps>
      </queue>
      <queue name="development">
         <maxResources>2048 mb, 0 vcores</maxResources><!--This parameter is for reference only.-->
         <aclSubmitApps>admin</aclSubmitApps>
      </queue>
      <queue name="production">
         <maxResources>7168 mb, 0 vcores</maxResources><!--This parameter is for reference only.-->
         <aclSubmitApps>omm</aclSubmitApps>
      </queue>
   </queue>
   <queuePlacementPolicy>
      <rule name="specified" create="false"/>
      <rule name="default" />
   </queuePlacementPolicy>
</allocations>
```

3. Open the **llama-site.xml** file and add the following configurations:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
 <property>
  <name>llama.am.throttling.maximum.placed.reservations.root.default</name>
  <value>1</value>
 </property>
 <property>
  <name>llama.am.throttling.maximum.queued.reservations.root.default</name>
  <value>2</value><!--This parameter is for reference only.-->
 </property>
 <property>
  <name>impala.admission-control.pool-default-query-options.root.default</name>
  <value>mem_limit=128m,query_timeout_s=20,max_io_buffers=10</value>
 </property>
 <property>
  <name>impala.admission-control.pool-queue-timeout-ms.root.default</name>
  <value>30000</value><!--This parameter is for reference only.-->
 </property>
```

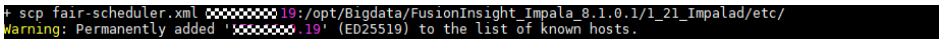


```
<property>
  <name>impala.admission-control.max-query-mem-limit.root.default</name>
  <value>3072000000</value><!--3GB--><!--For reference only-->
</property>
<property>
  <name>impala.admission-control.min-query-mem-limit.root.default</name>
  <value>2048000000</value><!--2GB-->
</property>
<property>
  <name>impala.admission-control.clamp-mem-limit-query-option.root.default.regularPool</name>
  <value>true</value>
 </property>
</configuration>
```

4. Run the following commands to synchronize **fair-scheduler.xml** and **llama-site.xml** to the **etc** folder in the installation directory on all Impalad nodes, respectively:

   scp fair-scheduler.xml {IP address of the Impalad instance}:/opt/Bigdata/FusionInsight_Impala_*/***_Impalad/etc/**

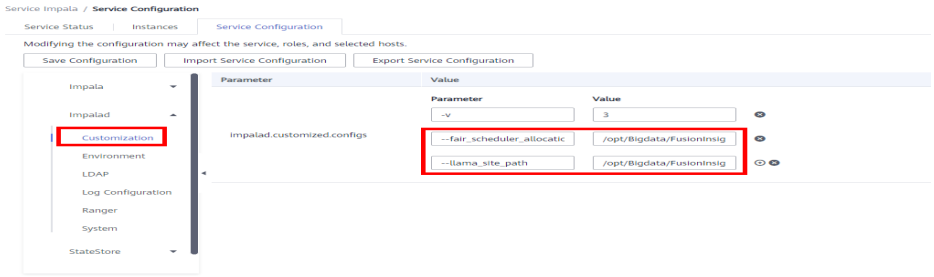
   scp llama-site.xml {IP address of the Impalad instance}:/opt/Bigdata/FusionInsight_Impala_*/***_Impalad/etc/**

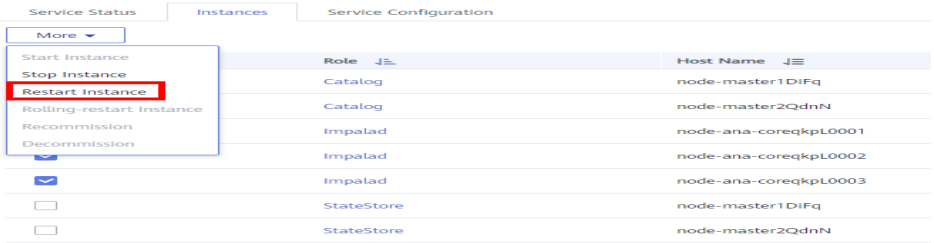
5. Log in to FusionInsight Manager, find the Impala component, and add the following custom configuration items and values to the Impala instance:

   --fair_scheduler_allocation_path The value is, for example, /opt/Bigdata/FusionInsight_Impala_*/***_Impalad/etc/fair-scheduler.xml**

   --llama_site_path The value is, for example, /opt/Bigdata/FusionInsight_Impala_*/***_Impalad/etc/llama-site.xml**

6. Restart the Impalad instance.

7. Log in to the node where the Impala client is installed, run the **source** command to obtain the environment variables, and run the following command:

   impala-shell -i {IP address of the Impalad instance:Port number} -Q request_pool=root.default (resource pool configured in fair-scheduler.xml and llama-site.xml)

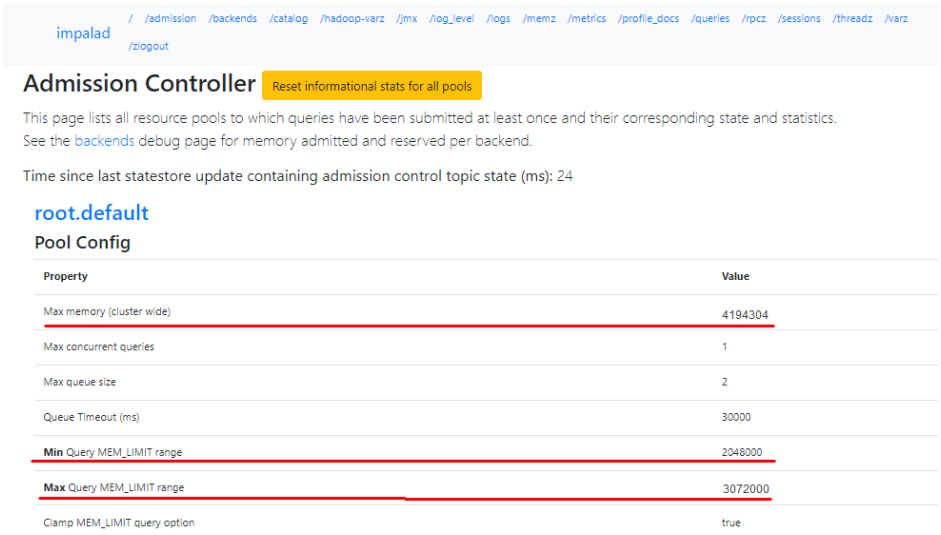

```
[root@node-master1IoKo ~]# impala-shell -i [redacted].19:21000 -Q request_pool=root.default
Starting Impala Shell without Kerberos authentication
Opened TCP connection to [redacted].19:21000
Connected to [redacted].19:21000
Server version: impalad version 3.4.0-RELEASE RELEASE (build ac0f95df4baa94fcfdc36ef370f6a432d582ac1f)
***********************************************************************************
Welcome to the Impala shell.
(Impala Shell v3.4.0-RELEASE (f68c12e) built on Sat Jun 26 17:16:01 CST 2021)

The '-B' command line flag turns off pretty-printing for query results. Use this
flag to remove formatting from results you want to save for later, or to benchmark
Impala.
***********************************************************************************
[[redacted].19:21000] default>
```

Execute SQL statements.

```
[[redacted]19:21000] default> select * from test1;
Query: select * from test1
Query submitted at: 2022-05-12 10:01:01 (Coordinator: http://[redacted].19:25000)
Query progress can be monitored at: http://[redacted].19:25000/query_plan?query_id=97440454dbab28ea:35bd90d600000000
```

8. Log in to the Impalad web UI to check the resource pool usage and apply the configurations.

   https://{*Cluster console address*}:9022/component/Impala/Impalad/95/



15.6 Using the Impala Query Management Page

Scenario

You can view Impala tasks through interactive queries on FusionInsight Manager.

NOTE

This section applies only to MRS 3.1.5 or later.

Prerequisites

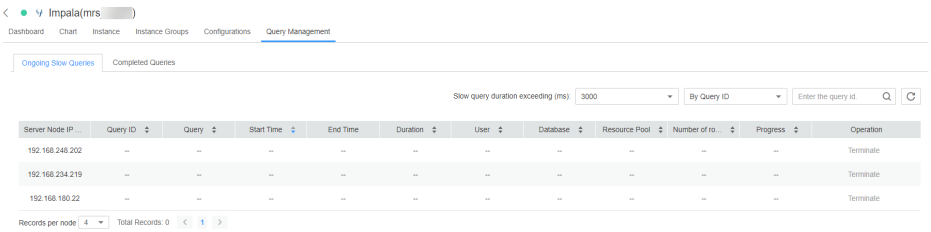
You have obtained the password of the **admin** user. The password of user **admin** is specified by the user during MRS cluster creation.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > *Name of the desired cluster* > **Service** > **Impala**.

Step 2 Click **Query Management**. By default, the list displays all ongoing queries.

Click **Completed Queries** to view the information about completed queries.



NOTE

You can search for slow queries by running duration, query ID, user, and database.

You can click **Stop** to stop an ongoing query task.

----End

15.7 Typical Impala Configurations

This section applies to MRS 3.*x* or later.

Page Access

On Manager, choose **Cluster** > **Services** > **Impala**. On the page that is displayed, click the **Configuration** tab then the **All Configurations** sub-tab. Enter a parameter name in the search box.

Parameter Description

NOTE

The following table lists only some common parameters. The actual parameters are subject to the FusionInsight Manager page. For details, visit the official website at **https://docs.cloudera.com/documentation/enterprise/6/properties/6.3/topics/cm_props_cdh630_impala.html**.

Table 15-3 Common Impala parameters

Parameter	Description	Default Value	Value Range
impalad.customized.configs	Custom configuration item of the impalad process	N/A	N/A
--enable_ldap_auth	Whether to enable LDAP authentication	false	The value can be **true** or **false**.

Parameter	Description	Default Value	Value Range
--ldap_bind_pattern	ldap userDNPattern, for example, **cn=%s,ou=People,dc=huawei,dc=com**	N/A	N/A
--ldap_passwords_in_clear_ok	If this parameter is set to **true**, LDAP passwords are sent in plaintext (excluding TLS and SSL) on the network.	false	The value can be **true** or **false**.
--ldap_uri-ip	ldap ip	N/A	N/A
--ldap_uri-port	ldap port	389	N/A
--max_log_files	Maximum number of process log files	10	N/A
--max_log_size	Maximum size of a process log file, in MB	200	N/A
statestored.customized.configs	Custom configuration item of the Statestored process	N/A	N/A
catalogd.customized.configs	Custom configuration item of the Catalogd process	N/A	N/A

15.8 Impala FAQ

15.8.1 Does Impala Support Disk Hot Swapping?

Question

Does Impala Support Disk Hot Swapping?

Answer

Impala data is stored in HDFS or OBS and does not need to be stored on local disks. Data only needs to be overflowed to disks (specified by **--scratch_dirs**) if memory space is not enough for service queries running on Impalad instances. Disk hot swapping is not supported. Impalad does not store temporary data in multiple copies.

16 Using IoTDB

16.1 Data Types and Encodings Supported by IoTDB

IoTDB supports the following data types and encodings. For details, see **Table 16-1**.

Table 16-1 Data types and encodings supported by IoTDB

Type	Description	Supported Encoding
BOOLEAN	Boolean	PLAIN, RLE
INT32	Integer	PLAIN, RLE, TS_2DIFF, GORILLA, FREQ, ZIGZAG
INT64	Long integer	PLAIN, RLE, TS_2DIFF, GORILLA, FREQ, ZIGZAG
FLOAT	Float	PLAIN, RLE, TS_2DIFF, GORILLA, FREQ
DOUBLE	Double	PLAIN, RLE, TS_2DIFF, GORILLA, FREQ
TEXT	String	PLAIN, DICTIONARY

16.2 IoTDB User Permission Management

16.2.1 IoTDB User Permission Description

MRS supports users, user groups, and roles. Permissions must be assigned to roles and then roles are bound to users or user groups. Users can obtain permissions only by binding a role or joining a group that is bound with a role.

NOTE

In security mode, you need to manage IoTDB permissions and add the created user to the **iotdbgroup** user group. In normal mode, IoTDB permission management is not required.

IoTDB Permission List

The **Name** column in **Table 16-2** lists the permissions supported by open-source IoTDB. If an MRS user needs to use corresponding permissions to perform operations, grant the permissions to the user on Manager by referring to the **Required Permission** column. For details, see **Creating an IoTDB Permission Role**.

Table 16-2 IoTDB permissions

Name	Description	Required Permission	Example
SET_STORAGE_GROUP	Used for creating a storage group, including setting permissions for the storage group and setting or canceling its time to live (TTL).	Set Storage Group	• set storage group to root.ln; • set ttl to root.ln 3600000; • unset ttl to root.ln;
CREATE_TIMESERIES	Used for creating a time series.	Create	• Creating a time series create timeseries root.ln.wf02.status with datatype=BOOLEAN,encoding=PLAIN; • Creating an aligned time series create aligned timeseries root.ln.device1(latitude FLOAT encoding=PLAIN compressor=SNAPPY, longitude FLOAT encoding=PLAIN compressor=SNAPPY);
INSERT_TIMESERIES	Used for inserting data.	Write	• insert into root.ln.wf02(timestamp,status) values(1,true); • insert into root.sg1.d1(time, s1, s2) aligned values(1, 1, 1);

Name	Description	Required Permission	Example
ALTER_TIMESERIES	Used for modifying a time series, and adding attributes and tags.	Alter	• alter timeseries root.turbine.d1.s1 ADD TAGS tag3=v3, tag4=v4; • ALTER timeseries root.turbine.d1.s1 UPSERT ALIAS=newAlias TAGS(tag2=newV2, tag3=v3) ATTRIBUTES(attr3=v3, attr4=v4);
READ_TIMESERIES	Used for querying data.	Read	• show storage group; • show child paths root.ln, show child nodes root.ln; • show devices; • show timeseries root.**; • show all ttl; • Querying data select * from root.ln.**; • Querying performance tracing tracing select * from root.**; • Querying the UDF select example(*) from root.sg.d1; • Querying statistics count devices;
DELETE_TIMESERIES	Used for deleting data or time series.	Delete	• Deleting a time series delete timeseries root.ln.wf01.wt01.status; • Deleting data delete from root.ln.wf02.wt02.status where time < 10;
DELETE_STORAGE_GROUP	Used for deleting a storage group.	IoTDB Admin Privilege	delete storage group root.ln;
CREATE_FUNCTION	Used for registering a UDF.	IoTDB Admin Privilege	create function example AS 'org.apache.iotdb.udf.UDTFExample';
DROP_FUNCTION	Used for deregistering a UDF.	IoTDB Admin Privilege	drop function example;

Name	Description	Required Permission	Example
UPDATE_TEMPLATE	Used for creating, deleting, and modifying metadata templates.	IoTDB Admin Privilege	create schema template t1(s1 int32);
READ_TEMPLATE	Used for viewing all metadata templates and metadata template content.	IoTDB Admin Privilege	• show schema templates; • show nodes in template t1;
APPLY_TEMPLATE	Used for attaching, detaching, and activating a metadata template.	IoTDB Admin Privilege	• set schema template t1 to root.sg.d; • create timeseries of schema template on root.sg.d;
READ_TEMPLATE_APPLICATION	Used for viewing the path for attaching or activating the metadata template.	IoTDB Admin Privilege	• show paths set schema template t1; • show paths using schema template t1;

16.2.2 Creating an IoTDB Permission Role

Create and configure an IoTDB role on Manager as an MRS cluster administrator. An IoTDB role can be configured with IoTDB administrator permissions or a common user's permissions to read, write, or delete data.

Prerequisites

- The MRS cluster administrator has understood service requirements.
- You have installed the IoTDB client.

Procedure

Step 1 Log in to FusionInsight Manager, choose **System** > **Permission** > **Role**.

Step 2 On the displayed page, click **Create Role** and specify **Role Name** and **Description**.

Step 3 Configure **Configure Resource Permission**. For details, see **Table 16-3**.

IoTDB permissions:

- **Common User Privileges**: includes data operation permissions. Permissions on the IoTDB **root** directory, storage group, and any node path from a storage group to a time series can be granted selectively. The minimum permissions are read, write, modify, and delete permissions on the time series.
- **IoTDB Admin Privilege**: includes all permissions in **Table 16-2**.

Table 16-3 Configuring a role

Scenario	Role Authorization
Configuring the IoTDB administrator permission	In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **IoTDB** and select **IoTDB Admin Privilege**.
Configuring the permission for users to create storage groups	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **IoTDB** > **Common User Privileges**. 2. Select **Set StorageGroup** for the **root** directory. 3. A user with this permission can create storage groups in the **root** directory.
Configuring the permission for users to create time series	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **IoTDB** > **Common User Privileges**. 2. Select **Create** for the **root** directory. You will have the permission to create time series in all recursive paths in the **root** directory. 3. Click **root** to go to the storage group page and select the **Create** permission for the corresponding storage group. You will have the permission to create time series in all recursive paths in the storage group directory.

Scenario	Role Authorization
Configuring the permission for users to modify time series	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **IoTDB** > **Common User Privileges**. 2. Select **Alter** for the **root** directory. You will have the permission to modify time series in all recursive paths in the **root** directory. 3. Click **root** to go to the storage group page and select the **Alter** permission for the corresponding storage group. You will have the permission to modify time series in all recursive paths of the storage group. 4. Click the specified storage group to go the time series page and select the **Alter** permission for the corresponding time series. You will have the permission to modify the time series.
Configuring the permission for users to insert data into time series	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **IoTDB** > **Common User Privileges**. 2. Select **Insert** for the **root** directory. You will have the permission to insert data into the time series in all recursive paths in the **root** directory. 3. Click **root** to go to the storage group page and select the **Insert** permission for the corresponding storage group. You will have the permission to insert data into the time series in all recursive paths of the storage group. 4. Click the specified storage group to go the time series page and select the **Insert** permission for the corresponding time series. You will have the permission to insert data into the time series.
Configuring the permission for users to read data from time series	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **IoTDB** > **Common User Privileges**. 2. Select **Read** for the **root** directory. You will have the permission to read data from the time series in all recursive paths in the **root** directory. 3. Click **root** to go to the storage group page and select the **Read** permission for the corresponding storage group. You will have the permission to read data from the time series in all recursive paths of the storage group. 4. Click the specified storage group to go the time series page and select the **Read** permission for the corresponding time series. You will have the permission to read data from the time series.

Scenario	Role Authorization
Configuring the permission for users to delete time series	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **IoTDB** > **Common User Privileges**. 2. Select **Delete** for the **root** directory. You will have the permission to delete data or time series in all recursive paths in the **root** directory. 3. Click **root** to go to the storage group page and select the **Delete** permission for the corresponding storage group. You will have the permission to delete data or time series in all recursive paths of the storage group. 4. Click the specified storage group to go the time series page and select the **Delete** permission for the corresponding time series. You will have the permission to delete data from the time series or delete the time series.

----End

16.3 Using the IoTDB Client

Scenario

This section describes how to use the IoTDB client in the O&M or service scenario.

Prerequisites

- The client has been installed. For example, the installation directory is **/opt/client**. The client directory in the following operations is only an example. Change it based on the actual installation directory onsite.
- Service component users have been created by the MRS cluster administrator. In security mode, machine-machine users need to download the keytab file. A human-machine user must change the password upon the first login.

Procedure

Step 1 Log in to the node where the client is installed as the client installation user.

Step 2 Run the following command to go to the client installation directory:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Before logging in to the IoTDB client for the first time, perform the following steps to generate a client SSL certificate:

1. Run the following command to generate a client SSL certificate:

   keytool -noprompt -import -alias myservercert -file ca.crt -keystore truststore.jks

   After running this command, you are required to set a password.

2. Copy the generated **truststore.jks** file to the ***Client installation directory*/IoTDB/iotdb/conf** directory.

   cp truststore.jks *Client installation directory*/**IoTDB/iotdb/conf**

Step 5 Log in to the IoTDB client based on the cluster authentication mode.

- In security mode, run the following command to authenticate the user and log in to the IoTDB client:

  kinit *Component service user*

- Skip this step in normal mode.

Step 6 Run the following command to switch to the directory where the IoTDB client running script is stored:

cd /opt/client/IoTDB/iotdb/sbin

Step 7 If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), invoke the **alter-cli-password.sh** script to change the default password of the default user **root**.

sh alter-cli-password.sh *IP address of the IoTDBServer instance RPC port number*

NOTE

- To view the IP address of the IoTDBServer instance node, log in to FusionInsight Manager and choose **Cluster** > **Services** > **IoTDB** > **Instances**.
- The IoTDBServer RPC port can be configured in the **IoTDB_SERVER_RPC_PORT** parameter. The default ports are as follows:
  - The default open-source port number is **6667**.
  - The default customized port number is **22260**.

  Port customization/open source: When creating an LTS version cluster, you can set **Component Port** to **Open source** or **Custom**. If **Open source** is selected, the open source port is used. If **Custom** is selected, the customized port is used.
- The initial password of user root is **root** in versions earlier than MRS 3.3.0 and **Iotdb@123** in MRS 3.3.3.0 and later versions.

  The password must contain at least four characters in versions earlier than MRS 3.3.0 and at least eight characters in MRS 3.3.0 and later versions, and cannot contain spaces.

Step 8 Run the following command to log in to the client:

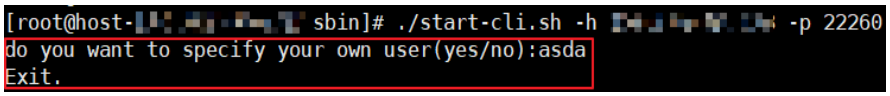
./start-cli.sh -h *IP address of the IoTDBServer instance node* **-p** *IoTDBServer RPC port*

After you run this command, specify the service username as required.

- To specify the service username, enter **yes** and enter the service username and password as prompted.

- If you will not specify the service username, enter **no**. In this case, you will perform subsequent operations as the user in **Step 5**.

- If you enter other information, you will log out.



NOTE

- If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), use the default user **root** to log in to the IoTDB client.
- When you log in to the client, you can configure the **-maxRPC** parameter to control the number of lines of execution results to be printed at a time. The default value is **1000**. If the value of **-maxRPC** is less than or equal to 0, all results are printed at a time. This parameter is typically used to redirect SQL execution results.
- Meanwhile, you can optionally use the **-disableISO8601** parameter to control the display format of the time column in the query result. If this parameter is not specified, the time is displayed in YYYYMMDDHHMMSS format. If this parameter is specified, the timestamp is displayed.
- If the SSL configuration is disabled on the server, you need to disable it on the client as follows:

  cd *Client installation directory*/**IoTDB/iotdb/conf**

  vi iotdb-client.env

  Change the value of **iotdb_ssl_enable** to **false**, save the configuration, and exit.

  To check the SSL configuration of the server, log in to FusionInsight Manager, choose **Cluster** > **Services** > **IoTDB** > **Configurations**, and search for **SSL_ENABLE**. Value **true** indicates that SSL is enabled, and value **false** indicates that it is disabled.

Step 9 After logging in to the client, you can run SQL statements.

----End

16.4 Getting Started with IoTDB

IoTDB is a data management engine that integrates collection, storage, and analysis of time series data. It features lightweight, high performance, and ease of use. It can connect to Hadoop and Spark ecosystems and meets the requirements

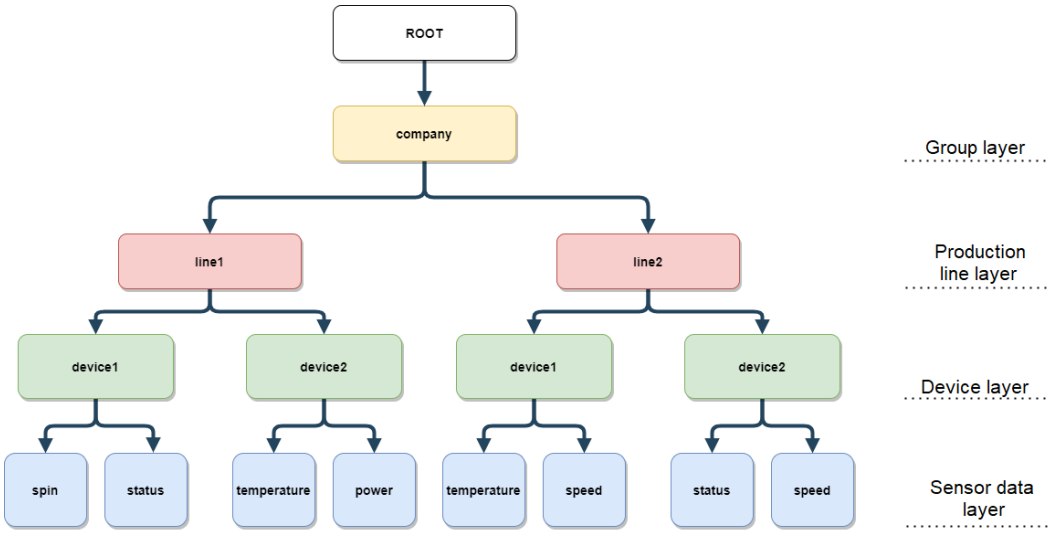
of high-speed write and complex analysis and query on massive time series data in industrial IoT applications.

Background

Assume that a group has three production lines with five devices on each. Sensors collect indicators (such as temperature, speed, and running status) of these devices in real time, as shown in **Figure 16-1**. The service process of storing and managing data using IoTDB is as follows:

1. Create a storage group named **root.***Group name* to represent the group.
2. Create time series to store the device indicators.
3. Simulate sensors and record indicators.
4. Run SQL statements to query indicators.
5. After the service is complete, delete the stored data.

Figure 16-1 Data structure



Procedure

Step 1 Log in to the client.

1. Log in to the node where the client is installed as the client installation user and run the following command to switch to the client installation directory, for example, **/opt/client**.

   cd /opt/client

2. Run the following command to configure environment variables:

   source bigdata_env

3. Before logging in to the IoTDB client for the first time, perform the following steps to generate a client SSL certificate:

   a. Run the following command to generate a client SSL certificate:

   keytool -noprompt -import -alias myservercert -file ca.crt -keystore truststore.jks

   After running this command, you are required to set a password.

b. Copy the generated **truststore.jks** file to the ***Client installation directory*/IoTDB/iotdb/conf** directory.

cp truststore.jks *Client installation directory*/**IoTDB/iotdb/conf**

4. If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the current user. The current user must have the permission to create IoTDB tables. For details, see **IoTDB User Permission Management**. If Kerberos authentication is disabled for the current cluster, skip this step.

kinit *MRS cluster user*

Example:

kinit iotdbuser

Step 2 Run the following command to switch to the directory where the script for running IoTDB client is stored:

cd /opt/client/IoTDB/iotdb/sbin

Step 3 If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), invoke the **alter-cli-password.sh** script to change the default password of the default user **root**.

sh alter-cli-password.sh *IP address of the IoTDBServer instance RPC port number*

NOTE

- To view the IP address of the IoTDBServer instance node, log in to FusionInsight Manager and choose **Cluster** > **Services** > **IoTDB** > **Instances**.
- The IoTDBServer RPC port can be configured in the **IoTDB_SERVER_RPC_PORT** parameter. The default ports are as follows:
  - The default open-source port number is **6667**.
  - The default customized port number is **22260**.

  Port customization/open source: When creating an LTS version cluster, you can set **Component Port** to **Open source** or **Custom**. If **Open source** is selected, the open source port is used. If **Custom** is selected, the customized port is used.
- The initial password of user root is **root** in versions earlier than MRS 3.3.0 and **Iotdb@123** in MRS 3.3.3.0 and later versions.

  The password must contain at least four characters in versions earlier than MRS 3.3.0 and at least eight characters in MRS 3.3.0 and later versions, and cannot contain spaces.

Step 4 Run the following command to log in to the client:

./start-cli.sh -h *IP address of the IoTDBServer instance node* **-p** *IoTDBServer RPC port*

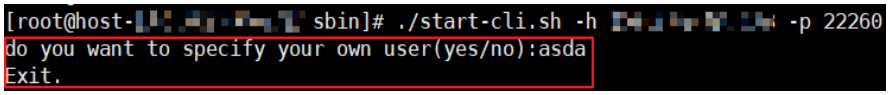
The default IoTDBServer RPC can be configured in the **IOTDB_SERVER_RPC_PORT** parameter.

After running this command, specify the service username as needed (user **root** is used for login when Kerberos authentication is disabled for the cluster (the cluster is in normal mode)).

- To specify the service username, enter **yes** and enter the service username and password as prompted.

- If you will not specify the service username, enter **no**. In this case, you will perform subsequent operations as the user in **Step 1.4**.

- If you enter other information, you will log out.



Step 5 Create a storage group/database (MRS 3.3.0 and later versions) named **root.company** based on **Figure 16-1**.

set storage group to root.company;

Step 6 Create corresponding time series for sensors of the devices on the production line.

create timeseries root.company.line1.device1.spin WITH DATATYPE=FLOAT, ENCODING=RLE;

create timeseries root.company.line1.device1.status WITH DATATYPE=BOOLEAN, ENCODING=PLAIN;

create timeseries root.company.line1.device2.temperature WITH DATATYPE=FLOAT, ENCODING=RLE;

create timeseries root.company.line1.device2.power WITH DATATYPE=FLOAT, ENCODING=RLE;

create timeseries root.company.line2.device1.temperature WITH DATATYPE=FLOAT, ENCODING=RLE;

create timeseries root.company.line2.device1.speed WITH DATATYPE=FLOAT, ENCODING=RLE;

create timeseries root.company.line2.device2.speed WITH DATATYPE=FLOAT, ENCODING=RLE;

create timeseries root.company.line2.device2.status WITH DATATYPE=BOOLEAN, ENCODING=PLAIN;

Step 7 Adds data to time series.

insert into root.company.line1.device1(timestamp, spin) values (now(), 6684.0);

insert into root.company.line1.device1(timestamp, status) values (now(), false);

insert into root.company.line1.device2(timestamp, temperature) values (now(), 66.7);

insert into root.company.line1.device2(timestamp, power) values (now(), 996.4);

insert into root.company.line2.device1(timestamp, temperature) values (now(), 2684.0);

insert into root.company.line2.device1(timestamp, speed) values (now(), 120.23);

insert into root.company.line2.device2(timestamp, speed) values (now(), 130.56);

insert into root.company.line2.device2(timestamp, status) values (now(), false);

Step 8 Query indicators of all devices on the production line 1.

select * from root.company.line1.;**

```
+-----------------------------+-----------------------------+---------------------------------
+-------------------------------------+-------------------------------+
|                         Time|root.company.line1.device1.spin|root.company.line1.device1.status|
root.company.line1.device2.temperature|root.company.line1.device2.power|
+-----------------------------+-----------------------------+---------------------------------
+-------------------------------------+-------------------------------+
|2021-06-17T11:29:08.131+08:00|                       6684.0|                             null|
null|                         null|
|2021-06-17T11:29:08.220+08:00|                         null|                            false|
null|                         null|
|2021-06-17T11:29:08.249+08:00|                         null|                             null|
66.7|                         null|
|2021-06-17T11:29:08.282+08:00|                         null|                             null|
null|                        996.4|
+-----------------------------+-----------------------------+---------------------------------
+-------------------------------------+-------------------------------+
```

Step 9 Delete all device indicators on the production line 2.

delete timeseries root.company.line2.;**

Query the indicator data on production line 2. The result shows no indicator data exists.

select * from root.company.line2.;**

```
+----+
|Time|
+----+
+----+
Empty set.
```

----End

16.5 IoTDB UDFs

16.5.1 IoTDB UDF Overview

IoTDB provides multiple built-in functions and user-defined functions (UDFs) to meet users' computing requirements.

UDF Types

Table 16-4 lists the UDF types supported by IoTDB.

Table 16-4 UDF types

Type	Description
User-defined timeseries generating function (UDTF)	This type of function can take multiple time series as input and generate one time series, which can contain any number of data points.

UDTF

To write a UDTF, you need to inherit the **org.apache.iotdb.db.query.udf.api.UDTF** class and implement at least the **beforeStart** method and one **transform** method.

Table 16-5 describes all interfaces that can be implemented by users.

Table 16-5 Interface description

Interface Definition	Description	Mandatory
void validate(UDFParameterValidator validator) throws Exception	This method is used to validate **UDFParameters** and is executed before **beforeStart** is called.	No
void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) throws Exception	This is an initialization method used to call the user-defined initialization behavior before the UDTF processes the input data. Each time a user executes a UDTF query, the framework constructs a new UDF instance, and this method is called. It is called only once in the lifecycle of each UDF instance.	Yes

Interface Definition	Description	Mandatory
void transform(Row row, PointCollector collector) throws Exception	This method is called by the framework. When you choose to use the **RowByRowAccessStrategy** strategy in **beforeStart** to consume raw data, this data processing method is called. The input data is passed in by **Row**, and the result is output by **PointCollector**. You need to call the data collection method provided by **collector** in this method to determine the output data.	Use either this method or **transform(RowWindow rowWindow, PointCollector collector)**.
void transform(RowWindow rowWindow, PointCollector collector) throws Exception	This method is called by the framework. When you choose to use the **SlidingSizeWindowAccessStrategy** or **SlidingTimeWindowAccessStrategy** strategy in **beforeStart** to consume raw data, this data processing method will be called. The input data is passed in by **RowWindow**, and the result is output by **PointCollector**. You need to call the data collection method provided by **collector** in this method to determine the output data.	Use either this method or **transform(Row row, PointCollector collector)**.
void terminate(PointCollector collector) throws Exception	This method is called by the framework. This method is called after all **transform** calls have been executed and before **beforeDestory** is called. In a single UDF query, this method will be called only once. You need to call the data collection method provided by **collector** in this method to determine the output data.	No
void beforeDestroy()	This method is called by the framework after the last input data is processed, and will be called only once in the lifecycle of each UDF instance.	No

Calling sequence of each method:

1. **void validate(UDFParameterValidator validator) throws Exception**
2. **void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) throws Exception**
3. **void transform(Row row, PointCollector collector) throws Exception** or **void transform(RowWindow rowWindow, PointCollector collector) throws Exception**
4. **void terminate(PointCollector collector) throws Exception**

5. **void beforeDestroy()**

NOTICE

Each time the framework executes a UDTF query, a new UDF instance will be constructed. When the query ends, this UDF instance will be destroyed. Therefore, the internal data of the instances in different UDTF queries (even in the same SQL statement) is isolated. You can maintain some state data in the UDTF without considering the impact of concurrency and other factors.

Interface usage:

- void validate(UDFParameterValidator validator) throws Exception

  The **validate** method is used to validate the parameters entered by users.

  In this method, you can limit the number and types of input time series, check the attributes of user input, or perform any custom logic verification.
- void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) throws Exception

  Using this method, you can do the following things:
  - Use **UDFParameters** to get the time series paths and parse the entered key-value pair attributes.
  - Set information required for running the UDF. That is, set the strategy to access the raw data and set the output data type in **UDTFConfigurations**.
  - Create resources, such as creating external connections and opening files.

UDFParameters

UDFParameters is used to parse the UDF parameters in SQL statements (the part in the parentheses following the UDF name in the SQL statements). The parameters include two parts. The first part is the path and its data type of the time series to be processed by the UDF. The second part is the key-value pair attributes for customization.

Example:

```
SELECT UDF(s1, s2, 'key1'='iotdb', 'key2'='123.45') FROM root.sg.d;
```

Usage:

```
void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) throws Exception {
  // parameters
 for (PartialPath path : parameters.getPaths()) {
    TSDataType dataType = parameters.getDataType(path);
   // do something
  }
  String stringValue = parameters.getString("key1"); // iotdb
  Float floatValue = parameters.getFloat("key2"); // 123.45
  Double doubleValue = parameters.getDouble("key3"); // null
  int intValue = parameters.getIntOrDefault("key4", 678); // 678
  // do something

  // configurations
  // ...
}
```


UDTFConfigurations

You can use **UDTFConfigurations** to specify the strategy used by the UDF to access raw data and the type of the output time series.

Usage:

```
void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) throws Exception {
  // parameters
  // ...

  // configurations
  configurations
    .setAccessStrategy(new RowByRowAccessStrategy())
    .setOutputDataType(TSDataType.INT32);
}
```

The **setAccessStrategy** method is used to set the strategy used by the UDF to access raw data. The **setOutputDataType** method is used to set the data type of the output time series.

- setAccessStrategy

  Note that the raw data access strategy you set here determines which **transform** method the framework will call. Implement the **transform** method corresponding to the raw data access strategy. You can also dynamically decide which strategy to set based on the attribute parameters parsed by **UDFParameters**. Therefore, the two **transform** methods are also allowed to be implemented in one UDF.

  Table 16-6 is the strategies you can set.

  Table 16-6 Policies for accessing raw data

Interface Definition	Description	transform Method to Call
RowByRowAccessStrategy	Processes raw data row by row. The framework calls the **transform** method once for each row of raw data input. When a UDF has only one input time series, a row of input is a data point in the input time series. When a UDF has multiple input time series, a row of input is a result record of the raw query (aligned by time) on these input time series. (In a row, there may be a column with a value of **null**, but not all of them are **null**.)	void transform(Row row, PointCollector collector) throws Exception

Interface Definition	Description	transform Method to Call
SlidingTimeWindowAccessStrategy	Processes a batch of data in a fixed time interval each time. A data batch is called a window. The framework calls the **transform** method once for each raw data input window. A window may contain multiple rows of data. Each row of data is a result record of the raw query (aligned by time) on these input time series. (In a row, there may be a column with a value of **null**, but not all of them are **null**.)	void transform(RowWindow rowWindow, PointCollector collector) throws Exception
SlidingSizeWindowAccessStrategy	Processes raw data batch by batch, and each batch contains a fixed number of raw data rows (except the last batch). A data batch is called a window. The framework calls the **transform** method once for each raw data input window. A window may contain multiple rows of data. Each row of data is a result record of the raw query (aligned by time) on these input time series. (In a row, there may be a column with a value of **null**, but not all of them are **null**.)	void transform(RowWindow rowWindow, PointCollector collector) throws Exception

The construction of **RowByRowAccessStrategy** does not require any parameters.

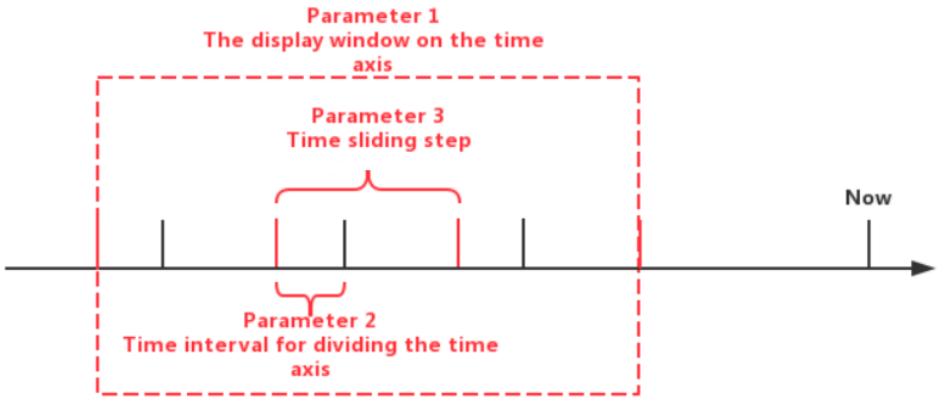
SlidingTimeWindowAccessStrategy has multiple constructors, and you can pass the following types of parameters to the constructors:

- Start time and end time of the display window on the time axis
- Time interval for dividing the time axis (must be positive)
- Time sliding step (not required to be greater than or equal to the time interval, but must be a positive number)

The display window on the time axis is optional. If these parameters are not provided, the start time of the display window will be set to the same as the minimum timestamp of the query result set, and the end time of the display window will be set to the same as the maximum timestamp of the query result set.

The sliding step parameter is also optional. If the parameter is not provided, the sliding step will be set to the same as the time interval for dividing the time axis.

The following figure shows the relationship between the three types of parameters.



Note that the actual time interval of some of the last time windows may be less than the specified time interval parameter. In addition, the number of data rows in some time windows may be 0. In this case, the framework will also call the **transform** method for the empty windows.

SlidingSizeWindowAccessStrategy has multiple constructors, and you can pass the following types of parameters to the constructors:

- Window size, that is, the number of data rows in a data processing window. Note that the number of data rows in some of the last time windows may be less than the specified number of data rows.
- Sliding step, that is, the number of rows between the first point of the next window and the first point of the current window. (This parameter is not required to be greater than or equal to the window size, but must be a positive number.)

The sliding step parameter is optional. If this parameter is not provided, the sliding step will be set to the same as the window size.

Note that the type of output time series you set here determines the type of data that **PointCollector** in the **transform** method can actually receive. The relationship between the output data type set in **setOutputDataType** and the actual data output type that **PointCollector** can receive is as follows.

Table 16-7 Mapping between the data types that can be received by the PointCollector and the output

Output Data Type Set in setOutputDataType	Data Type That PointCollector Can Receive
INT32	int
INT64	long
FLOAT	float
DOUBLE	double
BOOLEAN	boolean
TEXT	java.lang.String and org.apache.iotdb.tsfile.utils.Binary

- The type of the output time series of a UDTF is determined at runtime. The UDTF can dynamically determine the type of the output time series according to the type of the input time series.

  Example:

```
void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) throws Exception {
  // do something
  // ...

  configurations
    .setAccessStrategy(new RowByRowAccessStrategy())
    .setOutputDataType(parameters.getDataType(0));
}
```

  – void transform(Row row, PointCollector collector) throws Exception

    You need to implement this method when you specify the strategy for the UDF to read raw data as **RowByRowAccessStrategy** in **beforeStart**.

    This method processes one row of raw data at a time. The raw data is input from **Row** and output by **PointCollector**. You can choose to output any number of data points in one **transform** call. Note that the type of the output data points must be the same as you set in the **beforeStart** method, and the timestamp of the output data points must be strictly monotonically increasing.

    The following is a complete UDF example that implements the **void transform(Row row, PointCollector collector) throws Exception** method. It is an adder that receives two columns of time series as input. When two data points in a row are not **null**, this UDF will output the algebraic sum of these two data points.

```
import org.apache.iotdb.db.query.udf.api.UDTF;
import org.apache.iotdb.db.query.udf.api.access.Row;
import org.apache.iotdb.db.query.udf.api.collector.PointCollector;
import org.apache.iotdb.db.query.udf.api.customizer.config.UDTFConfigurations;
import org.apache.iotdb.db.query.udf.api.customizer.parameter.UDFParameters;
import org.apache.iotdb.db.query.udf.api.customizer.strategy.RowByRowAccessStrategy;
import org.apache.iotdb.tsfile.file.metadata.enums.TSDataType;

public class Adder implements UDTF {

  @Override
  public void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) {
    configurations
        .setOutputDataType(TSDataType.INT64)
        .setAccessStrategy(new RowByRowAccessStrategy());
  }

  @Override
  public void transform(Row row, PointCollector collector) throws Exception {
    if (row.isNull(0) || row.isNull(1)) {
      return;
    }
    collector.putLong(row.getTime(), row.getLong(0) + row.getLong(1));
  }
}
```

  – void transform(RowWindow rowWindow, PointCollector collector) throws Exception

    You need to implement this method when you specify the strategy for the UDF to read raw data as **SlidingTimeWindowAccessStrategy** or **SlidingSizeWindowAccessStrategy**.

    This method processes a batch of data in a fixed number of rows or a fixed time interval each time, and the container containing this batch of

data is called a window. The raw data is input from **RowWindow** and output by **PointCollector**. **RowWindow** can help you access a batch of rows, and it provides a set of interfaces for random access and iterative access to this batch of rows. You can choose to output any number of data points in one **transform** call. Note that the type of output data points must be the same as you set in the **beforeStart** method, and the timestamps of output data points must be strictly monotonically increasing.

The following is a complete UDF example that implements the **void transform(RowWindow rowWindow, PointCollector collector) throws Exception** method. It is a counter that receives any number of time series as input, and its function is to count and output the number of data rows in each time window within a specified time range.

```
import java.io.IOException;
import org.apache.iotdb.db.query.udf.api.UDTF;
import org.apache.iotdb.db.query.udf.api.access.RowWindow;
import org.apache.iotdb.db.query.udf.api.collector.PointCollector;
import org.apache.iotdb.db.query.udf.api.customizer.config.UDTFConfigurations;
import org.apache.iotdb.db.query.udf.api.customizer.parameter.UDFParameters;
import org.apache.iotdb.db.query.udf.api.customizer.strategy.SlidingTimeWindowAccessStrategy;
import org.apache.iotdb.tsfile.file.metadata.enums.TSDataType;

public class Counter implements UDTF {

 @Override
 public void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) {
   configurations
       .setOutputDataType(TSDataType.INT32)
       .setAccessStrategy(new SlidingTimeWindowAccessStrategy(
           parameters.getLong("time_interval"),
           parameters.getLong("sliding_step"),
           parameters.getLong("display_window_begin"),
           parameters.getLong("display_window_end")));
 }

 @Override
 public void transform(RowWindow rowWindow, PointCollector collector) throws Exception {
   if (rowWindow.windowSize() != 0) {
     collector.putInt(rowWindow.getRow(0).getTime(), rowWindow.windowSize());
   }
 }
}
```

- void terminate(PointCollector collector) throws Exception

  In some scenarios, a UDF needs to traverse all the raw data to calculate the final output data points. The **terminate** interface provides support for those scenarios.

  This method is called after all **transform** calls have been executed and before **beforeDestory** is called. You can implement the **transform** method to perform pure data processing, and implement the **terminate** method to output the processing results.

  The processing results need to be output by **PointCollector**. You can choose to output any number of data points in one **terminate** call. Note that the type of the output data points must be the same as you set in the **beforeStart** method, and the timestamp of the output data points must be strictly monotonically increasing.

  The following is a complete UDF example that implements the **void terminate(PointCollector collector) throws Exception** method. It takes

one time series whose data type is **INT32** as input, and outputs the maximum value point of the series.

```
import java.io.IOException;
import org.apache.iotdb.db.query.udf.api.UDTF;
import org.apache.iotdb.db.query.udf.api.access.Row;
import org.apache.iotdb.db.query.udf.api.collector.PointCollector;
import org.apache.iotdb.db.query.udf.api.customizer.config.UDTFConfigurations;
import org.apache.iotdb.db.query.udf.api.customizer.parameter.UDFParameters;
import org.apache.iotdb.db.query.udf.api.customizer.strategy.RowByRowAccessStrategy;
import org.apache.iotdb.tsfile.file.metadata.enums.TSDataType;

public class Max implements UDTF {

  private Long time;
  private int value;

  @Override
  public void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) {
    configurations
        .setOutputDataType(TSDataType.INT32)
        .setAccessStrategy(new RowByRowAccessStrategy());
  }

  @Override
  public void transform(Row row, PointCollector collector) {
    int candidateValue = row.getInt(0);
    if (time == null || value < candidateValue) {
      time = row.getTime();
      value = candidateValue;
    }
  }

  @Override
  public void terminate(PointCollector collector) throws IOException {
    if (time != null) {
      collector.putInt(time, value);
    }
  }
}
```

- void beforeDestroy()

  This method is used to terminate a UDF.

  This method is called by the framework. For a UDF instance, **beforeDestroy** will be called after the last record is processed. In the entire lifecycle of the instance, **beforeDestroy** will be called only once.

16.5.2 IoTDB UDF Sample Code and Operations

Complete UDF Sample Code

For details, see **IoTDB UDF Sample Code**.

Procedure

Step 1 Register a UDF.

Register a UDF with a full class name **com.huawei.bigdata.iotdb.UDTFExample** as follows:

1. Package the project into a JAR. If you use Maven to manage your project, you can refer to step "Build a JAR file" in following section:

- For details about clusters with Kerberos authentication enabled, see **Registering a UDF**.
- For details about how to disable Kerberos authentication for a cluster, see **Registering a UDF**

2. Log in to the node where IoTDBServer is located as user **root**, run **su - omm** to switch to user **omm**, and import the JAR package obtained in **Step 1.1** to the **$BIGDATA_HOME/FusionInsight_IoTDB_*/install/FusionInsight-IoTDB-*/iotdb/ext/udf** directory.

NOTICE

During cluster deployment, ensure that a corresponding JAR package exists in the UDF JAR package path of each IoTDBserver node. You can modify the IoTDB configuration **file udf_root_dir** to specify the root path for the UDF to load the JAR package.

3. Execute the following SQL statement to register the UDF:

   CREATE FUNCTION <UDF-NAME> **AS** '<UDF-CLASS-FULL-PATHNAME>'

   For example, to register a UDF named **example**, execute the following statement:

   ```
   CREATE FUNCTION example AS 'com.huawei.bigdata.iotdb.UDTFExample'
   ```

   Because IoTDB UDF instances are dynamically loaded through the reflection technology, you do not need to restart the server during the UDF registration process.

NOTICE

- UDF function names are case insensitive.
- Ensure that the function name given to the UDF is different from all built-in function names. A UDF with the same name as a built-in function cannot be registered.
- It is recommended that you do not use classes that have the same class name but different function logic in different JAR packages. For example, in **UDF(UDAF/UDTF): udf1, udf2**, the JAR package of **udf1** is **udf1.jar** and the JAR package of **udf2** is **udf2.jar**. Assume that both JAR packages contain the **com.huawei.bigdata.iotdb.UDTFExample** class. If you use two UDFs in the same SQL statement at the same time, the system will randomly load either of them and may cause inconsistency in UDF execution behavior.

Step 2 Query the UDF.

- Basic SQL syntax supported:
  - SLIMIT / SOFFSET
  - LIMIT / OFFSET
  - NON ALIGN
  - Queries with value filters

  - Queries with time filters
- Queries with aligned time series

  Currently, aligned time series are not supported in UDF queries. An error message is reported if you use UDF queries with aligned time series selected.
- Queries with an asterisk (*) in **SELECT** clauses

  Assume that there are two time series (**root.sg.d1.s1** and **root.sg.d1.s2**) in the system.
  - Execute **SELECT example(*) from root.sg.d1**.

    Then the result set will include the results of **example (root.sg.d1.s1)** and **example (root.sg.d1.s2)**.
  - Execute **SELECT example(s1, *) from root.sg.d1**.

    Then the result set will include the results of **example(root.sg.d1.s1, root.sg.d1.s1)** and **example(root.sg.d1.s1, root.sg.d1.s2)**.
  - Execute **SELECT example(*, *) from root.sg.d1**.

    Then the result set will include the results of **example(root.sg.d1.s1, root.sg.d1.s1)**, **example(root.sg.d1.s2, root.sg.d1.s1)**, **example(root.sg.d1.s1, root.sg.d1.s2)**, and **example(root.sg.d1.s2, root.sg.d1.s2)**.
- Queries with key-value pair attributes in UDF parameters

  You can pass any number of key-value pair parameters to the UDF when constructing a UDF query. The key and value in the key-value pair need to be enclosed in single or double quotes. Note that key-value pair parameters can be passed in only after all time series have been passed in. Example:

  ```
  SELECT example(s1, 'key1'='value1', 'key2'='value2'), example(*, 'key3'='value3') FROM root.sg.d1;
  SELECT example(s1, s2, 'key1'='value1', 'key2'='value2') FROM root.sg.d1;
  ```
- Showing all registered UDFs

  SHOW FUNCTIONS

Step 3 Deregister the UDF.

The following shows the SQL syntax of how to deregister a UDF:

DROP FUNCTION <UDF-NAME>

To deregister the UDF named **example**, execute the following statement:

```
DROP FUNCTION example
```

----End

16.6 IoTDB Performance Tuning

Scenario

You can increase IoTDB memory to improve IoTDB performance because read and write operations are performed in HBase memory.

Configuration

Log in to Manager, choose **Cluster** > **Services** > **IoTDB**, and click the **Configurations** tab and then **All Configurations**. Search the parameters and modify their values.

For details, see **Table 16-8**.

Table 16-8 IoTDB performance optimization parameters

Parameter	Description	Default Value	Optimization Suggestion
SSL_ENABLE	Whether to encrypt the channel between the client and server using SSL	true	**true** indicates that SSL encryption is enabled, and **false** indicates that SSL encryption is disabled. Data encryption and decryption during transmission have a great impact on performance. The test result shows that the performance gap is 200%. Therefore, you are advised to disable SSL encryption during the performance test. The parameter for the ConfigNode and IoTDBServer roles must be both modified.
iotdb_server_kerberos_qop	Encrypted data transmission of each IoTDBServer instance in the cluster. This parameter is supported only by clusters with Kerberos authentication enabled.	auth-int	**auth-int** indicates that data transmission is encrypted, and **auth** indicates that data is authenticated only without being encrypted. Therefore, you are advised to set this parameter to **auth**. The parameter for the ConfigNode and IoTDBServer roles must be both modified.

Parameter	Description	Default Value	Optimization Suggestion
GC_OPTS	Memory and garbage collection (GC) configuration parameters used by IoTDBServer. You need to set this parameter as required.	**-Xms2G -Xmx2G -XX:MaxDirectMemorySize=512M -XX:+UseG1GC -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -Djdk.tls.ephemeralDHKeySize=3072**	• **-Xms2G -Xmx2G** indicates the IoTDB JVM heap memory. Set this parameter to a large value in the scenarios with a large number of time series and concurrent writes. You can adjust the parameter value based on the GC duration threshold alarm or heap memory threshold alarm. If an alarm is generated, increase the parameter value by 0.5 times. If this alarm is frequently generated, double the value. When you adjust **HeapSize**, set **Xms** and **Xmx** to the same value to avoid performance deterioration during dynamic heap size adjustment by JVM. • **-XX:MaxDirectMemorySize** indicates the IoTDB JVM direct memory. The recommended value is 1/4 of the heap memory. This parameter mainly affects the write performance. If the write performance deteriorates significantly, you can increase the parameter value by 0.5 times. Note that the sum of the heap memory and direct memory must be less than or equal to 80% of the available system memory. Otherwise, IoTDB fails to be started. • Query scenario optimization example: If the query range is large, for example, a single time series contains more than 10,000 data points, the

Parameter	Description	Default Value	Optimization Suggestion
			quotient of 20% of the JVM memory allocated divided by the number of time series is recommended to be bigger than 160 KB for better performance of the storage engine in the default configuration. • For example, if there are 5 million sequences, the memory configuration is **-Xms128G -Xmx128G**.
write_read_schema_free_memory_proportion (MRS 3.3.0 and later versions: **storage_query_schema_consensus_free_memory_proportion**)	Memory allocation ratio: write, read, schema, and free	• MRS 3.2.0: 4:3:1:2 • MRS 3.3.0 and later versions: 3:3:1:1:2	You can adjust the memory based on the load. • A larger write memory means the better write throughput and single query performance. • A larger read memory means more supported concurrent queries. • A larger metadata memory means a lower probability of error message "IoTDB system load is too large". • A larger free memory means a lower probability of memory exhaustion.
iot_consensus_throttle_threshold_in_byte	Maximum size of the WAL directory, in bytes. The default maximum size is 50 GB. If the directory size exceeds the value you set, write operations are slowed down or rejected. Only MRS 3.3.0 and later versions support this function.	53687091200	You can adjust the value based on the number of writes. • For a small number of concurrent writes, do not change the value. • If there are a large number of concurrent writes, increase the value.

Parameter	Description	Default Value	Optimization Suggestion
data_region_iot_max_pending_batches_num	Maximum number of concurrent requests for synchronizing leader data copies to followers. Only MRS 3.3.0 and later versions support this function.	12	You can adjust the value based on the CPU usage and pending WAL files. To use this parameter, select **IoTDBServer(Role)** > **Customization** and add this parameter and its value to **engine.customized.configs**. ● For a small number of concurrent writes, do not change the value. ● If there are a large number of concurrent writes, increase the value. ● If there are a large number of pending WAL files, decrease the value. ● If the CPU usage is greater than 80% for a long time, decrease the value.
avg_series_point_number_threshold	Maximum average number of memory data points. When this threshold is reached, data is flushed to Tsfile. Only MRS 3.3.0 and later versions support this function.	10000	You can adjust the value based on the heap memory usage and GC duration. ● If the GC duration is long, decrease the value. ● If the memory usage is high, decrease the value.
flush_proportion	Write memory ratio for invoking disk flushing. If the write load is too high (for example, batch processing = 1000), you can decrease the value. Only MRS 3.3.0 and later versions support this function.	0.4	You can adjust the value based on the heap memory usage. If the memory usage is high, decrease the value.

16.7 IoTDB O&M Management

16.7.1 IoTDB Common Configuration Parameters

Scenario

IoTDB uses the multi-replica deployment architecture to implement cluster high availability. Each region (DataRegion and SchemaRegion) has three replicas by default. You can also configure more replicas. If a node is faulty, replicas on other nodes of the region replica can take over services from the faulty node, ensuring service continuity and improving cluster stability.

This section describes common configuration parameters for IoT database data operations.

Procedure

Step 1 Log in to Manager, choose **Cluster** > **Services** > **IoTDB** > **Configurations** > **All Configurations** to go to the IoTDB configuration page, and modify the parameters.

Step 2 Modify the ConfigNode and IoTDBServer configurations.

- Modifying the ConfigNode configuration:
  - Click **ConfigNode(Role)**. You can modify the existing configuration according to **Table 16-9**.
  - Choose **ConfigNode(Role)** > **Customization**. You can customize ConfigNode configurations in the **confignode.customized.configs** parameter according to **Table 16-9**.
- Modifying the IoTDBServer configuration:
  - Click **IoTDBServer(Role)**. You can modify the existing configuration according to **Table 16-9**.
  - Choose **IoTDBServer(Role)** > **Customization**. You can customize IoTDBServer configurations in the **engine.customized.configs** parameter according to **Table 16-9**.

Table 16-9 Common parameters

Parameter	Role	Example Value	Description
region_data_lost_proportion	ConfigNode	0.5	Region data supplementation starts when the lost data reaches the threshold (50% by default). **NOTE** This parameter is available only in MRS 3.3.0 or later.
region_repair_data_volume	ConfigNode	10	If the data volume in a region exceeds the threshold, the system automatically rectify the fault. The default value is **10** GB. **NOTE** This parameter is available only in MRS 3.3.0 or later.
dest_datanode_remaining_disk_space_proportion	ConfigNode	0.7	Percentage of the region data volume to the remaining disk space of the target DataNode when region replicas are supplemented. The default value is **70%**. **NOTE** This parameter is available only in MRS 3.3.0 or later.
read_consistency_level	ConfigNode	strong	Read consistency level. Currently, the value can only be **strong** or **weak**. In versions earlier than MRS 3.3.0, you need to set this parameter in the customized parameter **confignode.customized.configs**.
flush_proportion	IoTDBServer	0.4	Write memory ratio for invoking disk flushing. If the write load is too high (for example, batch processing = 1000), you can reduce the value.
replica_affinity_policy	IoTDBServer	random	When the value of **read_consistency_level** is **weak**, the strategy of the region replica node is selected for the query task.
coordinator_read_executor_size	IoTDBServer	20	Number of read thread cores of the IoTDBServer Coordinator of the custom parameter **engine.customized.configs**
rpc_thrift_compression_enable	ALL	false	Whether to compress data during transmission. Data is not compressed by default.
root.log.level	ALL	INFO	IoTDB log level. The modification of this parameter takes effect without restarting related instances.

Parameter	Role	Example Value	Description
SSL_ENABLE	ALL	true	Whether to encrypt the channel between the client and server using SSL

Step 3 Click **Save**.

Step 4 Click the **Instance** tab. Select the corresponding instance and choose **More** > **Restart Instance** to make the configuration take effect.

----End

16.7.2 IoTDB Log Overview

Log Description

Log paths: The default paths of IoTDB logs are **/var/log/Bigdata/iotdb/iotdbserver** (for storing run logs) and **/var/log/Bigdata/audit/iotdb/iotdbserver** (for storing audit logs).

Log archive rule: The automatic compression and archiving function of IoTDB is enabled. By default, when the size of a log file exceeds 20 MB (which is adjustable), the log file is automatically compressed. The naming rule of the compressed log file is as follows: <*Original log file name*>-<*yyyymmdd*>.*ID*.**log.gz**. A maximum of 10 latest compressed files are reserved. The number of compressed files and compression threshold can be configured.

Table 16-10 IoTDB log list

Type	Name	Description
Run logs	log-all.log	IoTDB service log.
	log-error.log	IoTDB service error log.
	log-measure.log	IoTDB service monitoring log.
	log-query-debug.log	IoTDB query debug log.
	log-query-frequency.log	IoTDB query frequency log.
	log-sync.log	IoTDB synchronization log.
	log-slow-sql.log	IoTDB slow SQL log.
	server.out	Log that records IoTDB service startup exceptions.
	postinstall.log	IoTDB process startup log.

Type	Name	Description
	prestart.log	Log that records IoTDB process startup exceptions.
	service-healthcheck.log	IoTDB database initialization log.
	start.log	IoTDBServer service startup log.
	stop.log	IoTDBServer service stop log.
	IoTDBServer-omm-<timestamp>-<pid>-gc.log.0.current	IoTDBServer service GC log.
Audit log	• MRS 3.2.0: log_audit.log • MRS 3.3.0 and later versions: log_datanode_audit.log	IoTDB audit log.

Log levels

Table 16-11 describes the log levels supported by IoTDB.

Levels of logs are ERROR, WARN, INFO, and DEBUG from the highest to the lowest priority. Run logs of equal or higher levels are recorded. The higher the specified log level, the fewer the logs recorded.

Table 16-11 Log levels

Level	Description
ERROR	Logs of this level record error information about system running.
WARN	Logs of this level record exception information about the current event processing.
INFO	Logs of this level record normal running status information about the system and events.
DEBUG	Logs of this level record the system information and system debugging information.

To modify log levels, perform the following operations:

1. Go to the **All Configurations** page of the IoTDB service by referring to **Modifying Cluster Service Configuration Parameters**.

2. In the navigation tree on the left, select **Log** corresponding to the role to be modified.
3. Select a desired log level and save the configuration.

NOTE

The IoTDB log level takes effect 60 seconds after being configured. You do not need to restart the service.

Log Formats

The following table lists the IoTDB log formats:

Table 16-12 Log formats

Type	Format	Example
Run log	<*yyyy-MM-dd HH:mm:ss,SSS*> | *Log level* | [*Thread name*] | *Log information* | *Log printing class* (*File*:*Line number*)	2021-06-08 10:08:41,221 | ERROR | [main] | Client failed to open SaslClientTransport to interact with a server during session initiation: | org.apache.iotdb.rpc.sasl.TFastSaslTransport (TFastSaslTransport.java:257)
Audit log	<*yyyy-MM-dd HH:mm:ss,SSS*> | *Log level* | [*Thread name*] | *Log information* | *Log printing class* (*File*:*Line number*)	2021-06-08 11:03:49,365 | INFO | [ClusterClient-1] | Session-1 is closing | IoTDB_AUDIT_LOGGER (TSServiceImpl.java:326)

16.7.3 Planning IoTDB Capacity

IoTDB has the multi-replica mechanism. By default, both schema regions and data regions have three replicas. The ConfigNode stores the mapping between regions and the IoTDBServer. The IoTDBServer stores region data and uses the file system of the OS to manage metadata and data files.

Capacity Specifications

- ConfigNode capacity specifications

  When a new storage group is created, IoTDB allocates 10,000 slots to it by default. When data is written, IoTDB allocates or creates a data region and mounts it to a slot based on the device name and time value. Therefore, the memory usage of a ConfigNode is related to the number of storage groups and the continuous write time of the storage groups.

Objects of Slot Allocation	Object Size (Byte)
TTimePartitionSlot	4
TSeriesPartitionSlot	8

Objects of Slot Allocation	Object Size (Byte)
TConsensusGroupId	4

According to the preceding table, creating one storage group that keeps running for 10 years requires about 0.68 GB of memory on a ConfigNode.

10,000 (slots) x 10 (years) x 53 (partitions) x (TTimePartitionSlot size + TSeriesPartitionSlot size + TConsensusGroupId size) = 0.68 GB

- IoTDBServer capacity specifications

  Data in IoTDB is allocated to IoTDBServers by region. By default, a region stores data as three replicas, and therefore three files are stored in the IoTDBServer file system. The upper limit of the IoTDBServer capacity is the maximum number of files that can be stored in the OS. For Linux, the upper limit is the number of inodes.

16.7.4 Manually Importing IoTDB Data

Scenario

This section describes how to use **import-csv.sh** to import data in CSV format to IoTDB.

Prerequisites

- The client has been installed. For details, see . For example, the installation directory is **/opt/client**. The client directory in the following operations is only an example. Change it based on the actual installation directory onsite.
- Service component users have been created by the MRS cluster administrator by referring to . In security mode, machine-machine users need to download the keytab file. For details, see . A human-machine user must change the password upon the first login.

Procedure

1. Prepare a CSV file named **example-filename.csv** on the local PC with the following content:

```
Time,root.fit.d1.s1,root.fit.d1.s2,root.fit.d2.s1,root.fit.d2.s3,root.fit.p.s1
1,100,hello,200,300,400
2,500,world,600,700,800
3,900,"hello, \"world\"",1000,1100,1200
```


NOTICE

Before importing data, pay attention to the following:

- In versions earlier than MRS 3.3.0, the data to be imported cannot contain spaces. Otherwise, importing that line of data fails and is skipped, but subsequent import operations are not affected.
- For MRS 3.3.0 or later, the data to be imported cannot contain spaces. Otherwise, the data fails to be imported. In this case, you need to check the type of the data to be imported.
- Data that contains commas (,) must be enclosed in backquote. For example, **hello,world** is changed to **`hello,world`**.
- Quotation marks ("") in the data must be replaced with the escape character **\"**. For example, **"world"** is changed to **\"world\"**.
- Single quotation marks (') in the data must be replaced with the escape character **\'**. For example, **'world'** will be changed to **\'world\'**.
- If the data to be imported is time, the format is **yyyy-MM-dd'T'HH:mm:ss**, **yyy-MM-dd HH:mm:ss** or **yyyy-MM-dd'T'HH:mm:ss.SSSZ**, for example, **2022-02-28T11:07:00**, **2022-02-28T11:07:00**, or **2022-02-28T11:07:00.000Z**.

2. Use WinSCP to import the CSV file to the directory of the node where the client is installed, for example, **/opt/client/IoTDB/iotdb/tools**.
3. Log in to the node where the client is installed as the client installation user.
4. Run the following command to switch to the client installation directory:

   cd /opt/client
5. Run the following command to configure environment variables:

   source bigdata_env
6. Before logging in to the IoTDB client for the first time, perform the following steps to generate a client SSL certificate:
   a. Run the following command to generate a client SSL certificate:

      keytool -noprompt -import -alias myservercert -file ca.crt -keystore truststore.jks

      After running this command, you are required to set a password.
   b. Copy the generated **truststore.jks** file to the ***Client installation directory*/IoTDB/iotdb/conf** directory.

      cp truststore.jks *Client installation directory*/**IoTDB/iotdb/conf**
7. (Optional) Perform this step to authenticate the current user if Kerberos authentication is enabled for the cluster. If Kerberos authentication is not enabled, skip this step.

   kinit *Component service user*
8. Run the following command to switch to the directory where the IoTDB client running script is stored:

   cd /opt/client/IoTDB/iotdb/sbin
9. If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), invoke the **alter-cli-password.sh** script to change the default password of the default user **root**.

sh alter-cli-password.sh *IP address of the IoTDBServer instance RPC port number*

NOTE

- To view the IP address of the IoTDBServer instance node, log in to FusionInsight Manager and choose **Cluster** > **Services** > **IoTDB** > **Instances**.
- The IoTDBServer RPC port can be configured in the **IoTDB_SERVER_RPC_PORT** parameter. The default ports are as follows:
  - The default open-source port number is **6667**.
  - The default customized port number is **22260**.

  Port customization/open source: When creating an LTS version cluster, you can set **Component Port** to **Open source** or **Custom**. If **Open source** is selected, the open source port is used. If **Custom** is selected, the customized port is used.
- The initial password of user root is **root** in versions earlier than MRS 3.3.0 and **Iotdb@123** in MRS 3.3.3.0 and later versions.

  The password must contain at least four characters in versions earlier than MRS 3.3.0 and at least eight characters in MRS 3.3.0 and later versions, and cannot contain spaces.

10. Run the following command to log in to the client:

    ./start-cli.sh -h *Service IP address of the IoTDBServer instance node* **-p** *IoTDBServer RPC port*

    NOTE

    - You can log in to FusionInsight Manager and choose **Cluster** > **Services** > **IoTDB** > **Instance** to view the service IP address of the IoTDBServer instance node.
    - To obtainthe default RPC port number, choose **Cluster** > **Services** > **IoTDB**, choose **Configurations** > **All Configurations**, and search for IOTDB_SERVER_RPC_PORT.
    - If Kerberos authentication is disabled for the cluster (the cluster is in normal mode), use the default user **root** to log in to the IoTDB client.

    After you run this command, specify the service username as required.

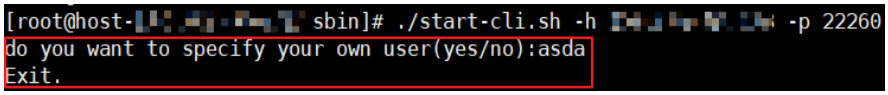
    – To specify the service username, enter **yes** and enter the service username and password as prompted.

    – If you will not specify the service username, enter **no**. In this case, you will perform subsequent operations as the user in **7**.

- If you enter other information, you will log out.



11. (Optional) Create metadata.

    IoTDB has the capability of type inference, so it is not necessary to create metadata before data import. However, it is recommended that you create metadata before using the CSV tool to import data, because this avoids unnecessary type conversion errors. The commands are as follows:

```
SET STORAGE GROUP TO root.fit.d1;
SET STORAGE GROUP TO root.fit.d2;
SET STORAGE GROUP TO root.fit.p;
CREATE TIMESERIES root.fit.d1.s1 WITH DATATYPE=INT32,ENCODING=RLE;
CREATE TIMESERIES root.fit.d1.s2 WITH DATATYPE=TEXT,ENCODING=PLAIN;
CREATE TIMESERIES root.fit.d2.s1 WITH DATATYPE=INT32,ENCODING=RLE;
CREATE TIMESERIES root.fit.d2.s3 WITH DATATYPE=INT32,ENCODING=RLE;
CREATE TIMESERIES root.fit.p.s1 WITH DATATYPE=INT32,ENCODING=RLE;
```

12. Run the following command to exit the client:

    quit;

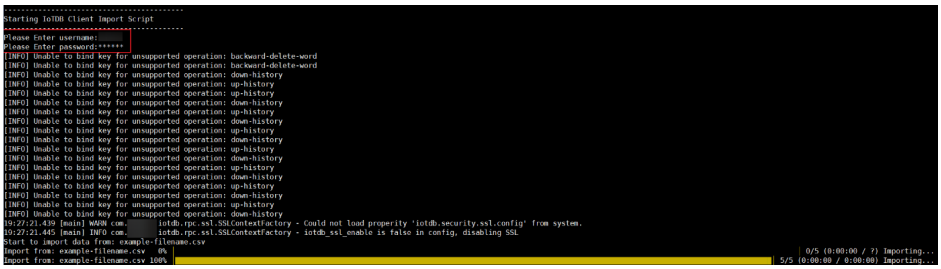
13. Run the following command to switch to the directory where the **import-csv.sh** script is stored:

    cd /opt/client/IoTDB/iotdb/tools

14. Run the following command to run **import-csv.sh** and import the **example-filename.csv** file:

    ./import-csv.sh -h *Service IP address of the IoTDBServer instance* **-p** *IoTDBServer RPC port* **-f** *example-filename.csv*

    Enter the service username and password in interactive mode as prompted. If information in the following figure is displayed, the CSV file is imported:



15. Verify data consistency.

    a. Run the following command to switch to the directory where the IoTDB client running script is stored:

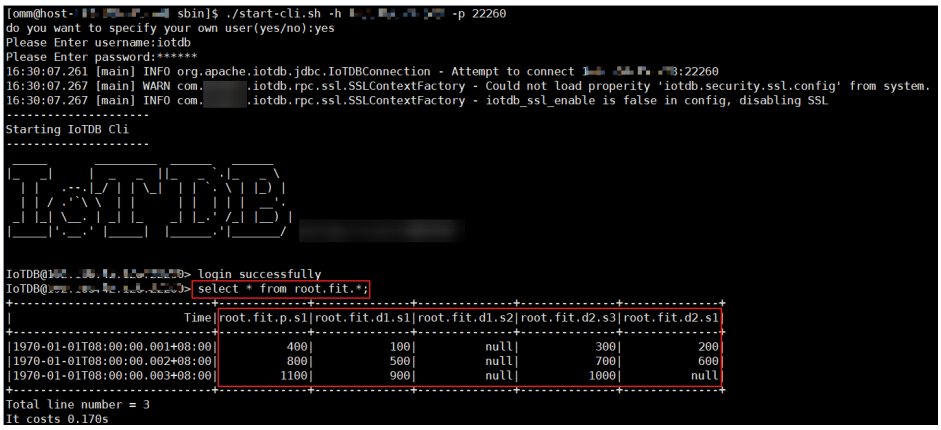
       cd /opt/client/IoTDB/iotdb/sbin

    b. Log in to the IoTDB client by referring to **10**. Run SQL statements to query data and compare the data with that in the **1** file.

    c. Check whether the imported data is consistent with the data in the **1**. If they are, the import is successful.

       Run the following command to check the imported data:

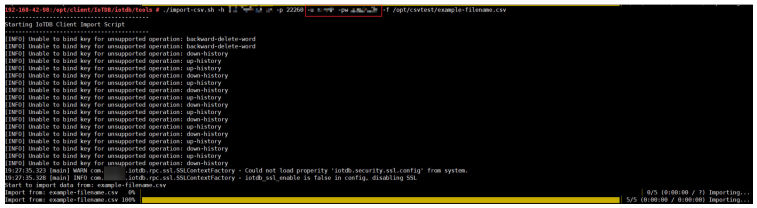
       SELECT * FROM root.fit.;**



NOTE

- To prevent security risks, you are advised to import CSV files in interactive mode.
- You can also import CSV files by running the **./import-csv.sh -h** *Service IP address of the IoTDBServer instance* **-p** *IoTDBServer RPC port* **-u** *Service username* **-pw** *Service user password***-f** *example-filename.csv* command.

  If information in the following figure is displayed, the CSV file is imported.



- If nanosecond (ns) time precision is enabled for the IoTDB on the server, the **-tp ns** parameter needs to be added when the client imports data with the nanosecond timestamp. To check whether nanosecond time precision is enabled for a cluster, log in to FusionInsight Manager, choose **Cluster** > **Configurations** > **All Non-default Values**, and search for **timestamp_precision**.

16.7.5 Manually Exporting IoTDB Data

Scenario

This section describes how to use **export-csv.sh** to export data from IoTDB to a CSV file.

NOTICE

Exporting data to CSV files may cause injection risks. Exercise caution when performing this operation.

Prerequisites

- The client has been installed. For details, see . For example, the installation directory is **/opt/client**. The client directory in the following operations is only an example. Change it based on the actual installation directory onsite.

- Service component users have been created by the MRS cluster administrator by referring to . In security mode, machine-machine users need to download the keytab file. For details, see . A human-machine user must change the password upon the first login.

Procedure

1. Log in to the node where the client is installed as the client installation user.
2. Run the following command to switch to the client installation directory:

   cd /opt/client
3. Run the following command to configure environment variables:

   source bigdata_env
4. Before logging in to the IoTDB client for the first time, perform the following steps to generate a client SSL certificate:

   a. Run the following command to generate a client SSL certificate:

      keytool -noprompt -import -alias myservercert -file ca.crt -keystore truststore.jks

      After running this command, you are required to set a password.

   b. Copy the generated **truststore.jks** file to the ***Client installation directory*/IoTDB/iotdb/conf** directory.

      cp truststore.jks *Client installation directory*/**IoTDB/iotdb/conf**
5. (Optional) Perform this step to authenticate the current user if Kerberos authentication is enabled for the cluster. If Kerberos authentication is not enabled, skip this step.

   kinit *Component service user*
6. Run the following command to switch to the directory where the **export-csv.sh** script is stored:

   cd /opt/client/IoTDB/iotdb/tools
7. Before executing the export script, input some queries or specify some SQL files. If a SQL file contains multiple SQL statements, the SQL statements must be separated by newline characters. For example:

   ```
   select * from root.fit.d1
   select * from root.sg1.d1
   ```
8. Execute **export-csv.sh** to export data.

   ./export-csv.sh -h *Service IP address of the IoTDBServer instance* **-p** *IoTDBServer RPC port* **-td <directory> [-tf <time-format> -s <sqlfile>]**

   Example:

   ```
   ./export-csv.sh -h x.x.x.x -p 22260 -td ./
   # Or
   ./export-csv.sh -h x.x.x.x -p 22260 -td ./ -tf yyyy-MM-dd\ HH:mm:ss
   # Or
   ./export-csv.sh -h x.x.x.x -p 22260 -td ./ -s sql.txt
   # Or
   ./export-csv.sh -h x.x.x.x -p 22260 -td ./ -tf yyyy-MM-dd\ HH:mm:ss -s sql.txt
   ```


NOTE

- You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **IoTDB**, and click the **Instance** tab to view the service IP address of the IoTDBServer instance node.
- The default RPC port number is **22260**. To obtain the default RPC port number, choose **Cluster** > **Services** > **IoTDB**, click the **Configurations** tab and then **All Configurations**, and search for IOTDB_SERVER_RPC_PORT.
- If the data exported contains a comma (,), it will be enclosed in double quotation marks (""). For example, **hello,world** is exported as **"hello,world"**.
- If the data exported contains double quotation marks (""), it will be enclosed in double quotation marks ("") and the original double quotation masks are replaced with **\"**. For example, **"world"** is exported as **"\"world\""**

9. When you run the command in **8**, a message is displayed indicating that CSV injection may occur. Enter **yes** to continue the command. If you enter another value, the data export operation will be canceled.

```
-------------------------------------------
Starting IoTDB Client Export Script
-------------------------------------------
Export data to CSV file may invoke CSV injection when opened in Windows.
Are you sure you want to continue(yes/no)?
```

For example, after you enter **yes**, enter the service username and password as prompted. If information in the following figure is displayed, the data is exported:

```
-------------------------------------------
Starting IoTDB Client Export Script
-------------------------------------------
Export data to CSV file may invoke CSV injection when opened in Windows.
Are you sure you want to continue(yes/no)?
yes
Please Enter username:[redacted]
Please Enter password:******
[INFO] Unable to bind key for unsupported operation: backward-delete-word
[INFO] Unable to bind key for unsupported operation: backward-delete-word
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
19:28:19.827 [main] WARN com.[redacted].iotdb.rpc.ssl.SSLContextFactory - Could not load properity 'iotdb.security.ssl.config' from system.
19:28:19.832 [main] INFO com.[redacted].iotdb.rpc.ssl.SSLContextFactory - iotdb_ssl_enable is false in config, disabling SSL
Start to export data from sql statement: select * from root.fit.d1
19:28:20.031 [main] DEBUG org.apache.iotdb.session.Session - EndPoint(ip:[redacted], port:22260) execute sql select * from root.fit.d1
Statement [select * from root.fit.d1] has dumped to file ./dump0.csv successfully! It costs 62ms to export 3 lines.
Start to export data from sql statement: select * from root.fit.d2
19:28:20.140 [main] DEBUG org.apache.iotdb.session.Session - EndPoint(ip:[redacted], port:22260) execute sql select * from root.fit.d2
Statement [select * from root.fit.d2] has dumped to file ./dump1.csv successfully! It costs 1ms to export 3 lines.
Start to export data from sql statement: select * from root.fit.p
19:28:20.175 [main] DEBUG org.apache.iotdb.session.Session - EndPoint(ip:[redacted], port:22260) execute sql select * from root.fit.p
Statement [select * from root.fit.p] has dumped to file ./dump2.csv successfully! It costs 1ms to export 3 lines.
```

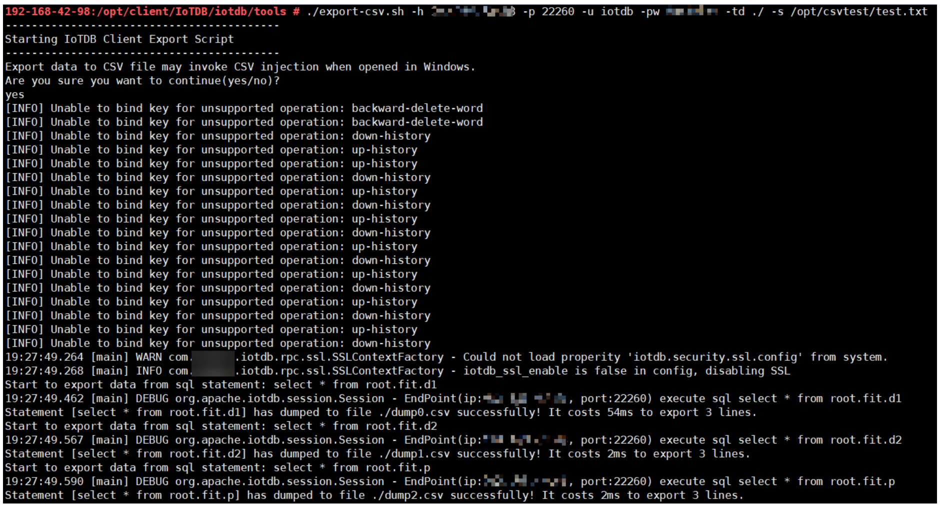
NOTE

- To prevent security risks, you are advised to export CSV files in interactive mode.
- You can also export CSV files by running the **./export-csv.sh -h** *Service IP address of the IoTDBServer instance* **-p** *IoTDBServer RPC port* **-u** *Service username* **-pw** *Service user password* **-td <directory> [-tf <time-format> -s <sqlfile>]** command. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

  Example:

  ```
  ./export-csv.sh -h x.x.x.x -p 22260 -u test -pw Password -td ./
  # Or
  ./export-csv.sh -h x.x.x.x -p 22260 -u test -pw Password -td ./
  # Or
  ./export-csv.sh -h x.x.x.x -p 22260 -u test -pw Password -td ./ -s sql.txt
  # Or
  ./export-csv.sh -h x.x.x.x -p 22260 -u test -pw Password -td ./ -tf yyyy-MM-dd\ HH:mm:ss -s sql.txt
  ```

  If information in the following figure is displayed, the CSV file is exported:

17 Using JobGateway

17.1 Configuring JobGateway Parameters

Page Access

Go to the JobGateway configuration page by referring to **Modifying Cluster Service Configuration Parameters**.

Parameters

Table 17-1 JobGateway parameters

Parameter	Description	Default Value
HTTP_INSTANCE_PORT	HTTP port of the JobServer service	Default value: **29973** Value range: 29970 to 29979
HTTPS_INSTANCE_PORT	HTTPS port of the JobServer service	Default value: **29972** Value range: 29970 to 29979
JAVA_OPTS	JVM parameter used for garbage collection (GC). Ensure that **GC_OPT** is set correctly. Otherwise, the process will fail to start.	See the default configuration on the page.
job.record.batch.delete.count	25	Number of aged data records in each batch of JobServers
job.record.expire.count	500000	Number of aged JobServer data records

Parameter	Description	Default Value
job.record.expire.day	7	Time when a JobServer job expires
logging.level.org.apache.tomcat	Log level of Tomcat logs on the JobServer server	Default value: **INFO** Value range: **DEBUG**, **INFO**, **WARN**, **ERROR**, or **FATAL**
root.level	Level of logs on the JobServer server	Default value: **INFO** Value range: **DEBUG**, **INFO**, **WARN**, **ERROR**, or **FATAL**
NGINX_PORT	Listening port of the JobBalancer service	Default value: The default HTTPS port number is **29970**, and the default HTTP port number is **29971**. Value range: 29970 to 29979
client_body_buffer_size	Sets the size of the buffer for reading the client request body. If the request body is larger than the buffer, the entire body or only part of it is written to the temporary file.	Default value: **10240** Value range: greater than 0
client_body_timeout	Defines the timeout interval for reading the client request body, in seconds. The timeout is set only for a period of time between two consecutive read operations, not for the transmission of the entire request body. If the client does not transmit any content within this period, the request is terminated and a 408 (request timeout) error occurs.	Default value: **60** Value range: 1 to 86400

Parameter	Description	Default Value
client_header_buffer_size	Sets the buffer size for reading client request headers. For most requests, a 1 KB of buffer is sufficient. However, if the request contains a long cookie or comes from a WAP client, 1KB may not be appropriate. If the request line or request header field is not suitable for this buffer, a larger buffer configured by the **large_client_header_buffers** directive is allocated.	Default value: **1024** Value range: greater than 0
client_header_timeout	Defines the timeout interval for reading the client request header. If the client does not transmit the entire header during this period, the request is terminated and a 408 (request timeout) error occurs.	Default value: **60** Value range: 1 to 86400
client_max_body_size	Maximum size of an HTTP request body, in MB	Default value: **80** Value range: 1 to 10240
keepalive_requests	Sets the maximum number of requests that can be served through a keepalive connection. After the maximum number of requests is sent, the connection is closed. Closing connections periodically is necessary to free up the memory allocation for each connection. Using a high maximum number of requests may cause excessive memory usage. Therefore, this method is not recommended.	Default value: **1000** Value range: 1 to 100000

Parameter	Description	Default Value
keepalive_time	Limits the maximum time that a request can be processed through a keepalive connection, in seconds. After the time specified by this parameter is reached, the connection is closed after subsequent requests are processed.	Default value: **3600** Value range: 1 to 86400
keepalive_timeout	Sets a timeout period during which keepalive client connections remain open on the server, in seconds. The zero value disables keepalive client connections.	Default value: **75** Value range: 0 to 86400
large_client_header_buffers.size	Sets the maximum number (**large_client_header_buffers.number**) and size of buffers used to read large client request headers. A request line cannot exceed the size of a buffer. Otherwise, error 414 (Request-URI Too Large) is returned to the client. The request header field cannot exceed the size of a buffer. Otherwise, error 400 (Bad Request) is returned to the client. The buffer is allocated only on demand. These buffers are released if the connection transitions to keep active after the request processing is complete.	Default value: **4096** Value range: greater than 0
lb_limit_req_burst	When a large number of requests are sent, the requests that exceed the access frequency limit are stored in the buffer, and error 503 is returned if the requests exceed the buffer size.	Default value: **50** Value range: 1 to 1000

Parameter	Description	Default Value
lb_limit_zone_rate	Rate limit on HTTP requests of clients with the same ID, in r/s or r/m. For example, **30r/s** indicates that 30 accesses are allowed per second.	Default value: **30r/s** Value range: 1 to 100r/s or 1 to 6000r/m
lb_limit_zone_size	Size of the HTTP memory buffer, in MB	Default value: **20** Value range: 1 to 10240
lb_req_timeout	Timeout interval for Nginx read/write	Default value: **60s** Value range: 1 to 3600s
proxy_connect_timeout	Defines the timeout interval for setting up a TCP connection with the proxy server. The value is a combination of numbers and units. m indicates minute and s indicates second.	Default value: **3m** Value range: 1 to 60 m or 1 to 3600s
proxy_timeout	Timeout between two consecutive read or write operations on a TCP connection to the proxy server. If no data is transmitted within this period, the connection is closed. The value is a combination of numbers and units. m indicates minute and s indicates second.	Default value: **3m** Value range: 1 to 60 m or 1 to 3600s

17.2 Manually Updating the Service Client of JobGateway

NOTE

This section applies to MRS 3.3.1 and later.

Scenario

If you enabled multi-service and the service client fails to be updated for the added services in the MRS cluster, you need to manually update the client.

Prerequisites

- A service has been added to Manager.
- The following procedure is available for updating clients of Spark, Hive, and Flink services.

Procedure

Step 1 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **JobGateway** > **Instances**, and view the IP addresses of all JobServer instance nodes.

Step 2 Obtain the node with the lowest IP address among the nodes where JobServer is deployed. If the service client fails to be updated on the node with the lowest IP address, go to **Step 3**. Otherwise, go to **Step 4**.

NOTE

- If the IP addresses of the nodes where JobServer is deployed are 192.168.0.192, 192.168.0.168, and 192.168.0.23, the lowest IP address is 192.168.0.23.
- The script can update one or more service clients at a time.

Step 3 Log in to the node with the lowest IP address as user **omm** and run the script for updating the service client.

cd ${BIGDATA_HOME}/FusionInsight_JobGateway_*/install/FusionInsight_JobGateway-*/adapter/script

dos2unix update-multi-service-client.sh

sh update-multi-service-client.sh *Service 1 Service 2*.....

Step 4 Log in to other nodes where the service client fails to be updated as user **omm** and run the following commands:

TIME_RECORD_FILE=${BIGDATA_HOME}/FusionInsight_JobGateway*/install/FusionInsight-JobGateway*/job-gateway/job-server/tmp/multi-service-update-time

date +"%Y-%m-%d %H:%M:%S" > "${TIME_RECORD_FILE}"

cat ${TIME_RECORD_FILE} (Ensure that the file is generated and the time is written.)

cd ${BIGDATA_HOME}/FusionInsight_JobGateway_*/install/FusionInsight_JobGateway-*/adapter/script

dos2unix update-multi-service-client.sh

sh update-multi-service-client.sh *Service 1 Service 2*.....

The following is an example:

Some nodes fail to be updated when the Hive-1 Spark-1 service client is updated. The node with the lowest IP address fails to be updated. The rectification is as follows:

- Run the following commands to update the service client script on the node with the lowest IP address:

  cd /opt/Bigdata/FusionInsight_JobGateway_8.5.0/install/FusionInsight-JobGateway-1.0.0/adapter/script

dos2unix update-multi-service-client.sh

sh update-multi-service-client.sh *Hive-1 Spark-1*

```
update-multi-service-client.sh: line 164: warning: command substitution: ignored null byte in input
Warning: Permanently added '192.168.234.117' (ED25519) to the list of known hosts.

Warning: Permanently added '192.168.234.117' (ED25519) to the list of known hosts.

Warning: Permanently added '192.168.234.243' (ED25519) to the list of known hosts.

Warning: Permanently added '192.168.234.243' (ED25519) to the list of known hosts.

Start to access REST API.
Warning: Permanently added '192.168.234.117' (ED25519) to the list of known hosts.

FusionInsight_Cluster_1_Services_Client.tar                 100% 1566MB 235.8MB/s
00:06
```

- Perform the following operations on other nodes where update fails:

  TIME_RECORD_FILE=${BIGDATA_HOME}/FusionInsight_JobGateway_8.5.0/install/FusionInsight-JobGateway-1.0.0/job-gateway/job-server/tmp/multi-service-update-time

  date +"%Y-%m-%d %H:%M:%S" > "${TIME_RECORD_FILE}"

  cat ${TIME_RECORD_FILE}

  ```
  2024-03-19 18:02:50
  ```

  cd /opt/Bigdata/FusionInsight_JobGateway_8.5.0/install/FusionInsight-JobGateway-1.0.0/adapter/script

  dos2unix update-multi-service-client.sh

  sh update-multi-service-client.sh *Hive-1 Spark-1*

  ```
  update-multi-service-client.sh: line 163: warning: command substitution: ignored null byte in input
  Warning: Permanently added '192.168.234.117' (ED25519) to the list of known hosts.

  Warning: Permanently added '192.168.234.117' (ED25519) to the list of known hosts.

  Warning: Permanently added '192.168.234.167' (ED25519) to the list of known hosts.

  Warning: Permanently added '192.168.234.167' (ED25519) to the list of known hosts.

  Warning: Permanently added '192.168.234.167' (ED25519) to the list of known hosts.

  FusionInsight_Cluster_1_Services_Client.tar                 100% 1566MB 378.2MB/s
  00:04
  Warning: Permanently added '192.168.234.167' (ED25519) to the list of known hosts.

  Warning: Permanently added '192.168.234.167' (ED25519) to the list of known hosts.
  ```

Step 5 Run the following commands on all JobServer nodes to delete the timestamps of all JobServer nodes:

TIME_RECORD_FILE=${BIGDATA_HOME}/FusionInsight_JobGateway*/install/FusionInsight-JobGateway*/job-gateway/job-server/tmp/multi-service-update-time

rm -f ${TIME_RECORD_FILE}

----End

17.3 JobGateway Log Overview

Log Description

Log path: **/var/log/Bigdata/job-gateway/**

Log archive rule: The automatic compression and archive function is enabled for JobGateway run logs. When the total size of all log files exceeds 20 MB (configurable), the log files are automatically compressed into a package named in the format of *Original log file name*-*yyyy-mm-dd*.*No.*.**log.zip**. A maximum of 20 latest compressed files are retained. The number of compressed files and compression threshold can be configured.

Table 17-2 JobGateway log list

Log Type	Log File Name	Description
JobServer run log	job-gateway.log	Service run log
	prestart.log	Service prestart log
	availability-check.log	Service availability check log
	verbose-gc-sp.txt	Service GC log
	gc.log	Service GC log
JobServer audit log	access_log.{yyyy-MM-dd}.log	Service audit log
Balance run log	availability-check.log	Service availability check log
	error.log	Service error log
	prestart.log	Service prestart log
	start.log	Service startup log
Balance audit log	access_http.log	Service audit log

Log Levels

The following table describes the log levels provided by JobGateway.

The log levels are ERROR, WARN, INFO, and DEBUG in descending order of priority. Only logs whose levels are higher than or equal to the specified level are recorded. The higher the log level specified, the fewer the logs are recorded.

Table 17-3 Log levels

Level	Description
ERROR	Logs of this level record error information about system running
WARN	Logs of this level record exception information about the current event processing
INFO	Logs of this level record normal running status information about the system and events
DEBUG	Logs of this level record the system information and system debugging information

To modify log levels, perform the following operations:

1. Log in to FusionInsight Manager.
2. Choose **Cluster** > **Services** > **JobGateway** and click **Configurations**.
3. Click **All Configurations**.
4. On the menu bar on the left, select the log menu of the target role.
5. Select a desired log level.
6. Click **Save** then **OK**.

Log Format

The following table lists the JobGateway log formats.

Table 17-4 Log formats

Log Type	Format	Example
Run log	<yyyy-MM-dd HH:mm:ss> <Log level> <Name of the thread that generates the log > <Name of the class that generates the log:Number of code lines >Message in the log >	[2024-05-22 10:37:10.000] [INFO] [job-status-refresh-task-fc1ff7bb-a49f-40bb-ada8-d6793c2bed20] [com.huawei.bigdata.jobgateway.job.task.JobStatusRefreshTask:60] - [start update job task]

Log Type	Format	Example
Audit log	<Remote host name > <Remote username > <Authenticated remote user > <yyyy-MM-dd HH:mm:ss,SSS > < "First line of the log request" Response code Number of sent bytes>	192.18.212.17 - - [29/Apr/2024:11:51:31 +0800] "GET /mrs/job/wait_nums HTTP/1.1" 200 11

18 Using Kafka

18.1 Kafka User Permission Management

18.1.1 Kafka User Permissions

Scenario

For clusters with Kerberos authentication enabled, using Kafka requires relevant permissions. MRS clusters can grant the use permission of Kafka to different users.

Table 18-1 lists the default Kafka user groups.

NOTE

Kafka supports two types of authentication plug-ins: Kafka open source authentication plug-in and Ranger authentication plug-in.

This section describes the user permission management based on the Kafka open source authentication plug-in. For details about how to use the Ranger authentication plug-in, see **Adding a Ranger Access Permission Policy for Kafka**.

Table 18-1 Default Kafka user groups

User Group	Description
kafkaadmin	Kafka administrator group. Users in this group have the permissions to create, delete, read, and write all topics, and authorize other users.
kafkasuperuser	Kafka super user group. Users in this group have the permissions to read and write all topics.
kafka	Kafka common user group. Users in this group can access a topic only when they are granted with the read and write permissions of the topic by a user in the **kafkaadmin** group.

Prerequisites

- You have installed the Kafka client.
- A user in the **kafkaadmin** group has been prepared.

Procedure

Step 1 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **ZooKeeper** > **Instances**.

Step 2 View the IP addresses of the ZooKeeper role instance.

Record the IP address of any ZooKeeper instance.

Step 3 Prepare the client based on service requirements. Log in to the node where the client is installed. For details, see **Using an MRS Client**.

Step 4 Run the following command to switch to the client installation directory, for example, **/opt/client/Kafka/kafka/bin**.

cd /opt/client/Kafka/kafka/bin

Step 5 Run the following command to configure environment variables:

source /opt/client/bigdata_env

Step 6 Run the following command to authenticate the user(skip this step in normal mode):

kinit *Component service user*

Step 7 he following lists the common **kafka-acl.sh** commands used for user authorization:

- View the permission control list of a topic:

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<Service IP address of any ZooKeeper node>***:2181/kafka** *>* **--list --topic** *<Topic name>*

  ./kafka-acls.sh --bootstrap-server *<IP address of the Kafkacluster:21007>* **--command-config ../config/client.properties --list --topic** *<topic name>*

- Add the Producer permission for a user:

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<Service IP address of any ZooKeeper node>***:2181/kafka > --add --allow-principal User:***<Username>* **--producer --topic** *<Topic name>*

  ./kafka-acls.sh --bootstrap-server *<IP address of the Kafkacluster:21007>* **--command-config ../config/client.properties --add --allow-principal User:***<username>* **--producer --topic** *<topic name>*

- Assign the Producer permission to a user in batches.

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<Service IP address of any ZooKeeper node>***:2181/kafka > --add --allow-principal User:***<Username>* **--producer --topic** *<Topic name>* **--resource-pattern-type prefixed**

  ./kafka-acls.sh --bootstrap-server *<IP address of the Kafkacluster:21007>* **--command-config ../config/client.properties --add --allow-principal User:***<username>* **--producer --topic** *<topic name>***--resource-pattern-type prefixed**

- Remove the Producer permission from a user:

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<Service IP address of any ZooKeeper node>***:2181/kafka > --remove --allow-principal User:***<Username>* **--producer --topic** *<Topic name>*

  ./kafka-acls.sh --bootstrap-server <*IP address of the Kafkacluster:21007*> **--command-config ../config/client.properties --remove --allow-principal User:***<username>* **--producer --topic** <*topic name*>

- Delete the Producer permission of a user in batches:

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<Service IP address of any ZooKeeper node>***:2181/kafka > --remove --allow-principal User:***<Username>* **--producer --topic** *<Topic name>* **--resource-pattern-type prefixed**

  ./kafka-acls.sh --bootstrap-server <*IP address of the Kafkacluster:21007*> **--command-config ../config/client.properties --remove --allow-principal User:***<username>* **--producer --topic** *<topic name>***--resource-pattern-type prefixed**

- Add the Consumer permission for a user:

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<Service IP address of any ZooKeeper node>***:2181/kafka > --add --allow-principal User:***<Username>* **--consumer --topic** *<Topicname>* **--group** *<Consumer group name>*

  ./kafka-acls.sh --bootstrap-server <*IP address of the Kafkacluster:21007*> **--command-config ../config/client.properties --add --allow-principal User:***<username>* **--consumer --topic** *<topicname>* **--group** *<consumer group name>*

- Add consumer permissions to a user in batches:

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<Service IP address of any ZooKeeper node>***:2181/kafka > --add --allow-principal User:***<Username>* **--consumer --topic** *<Topic name>* **--group** *<Consumer group name>* **--resource-pattern-type prefixed**

  ./kafka-acls.sh --bootstrap-server <*IP address of the Kafkacluster:21007*> **--command-config ../config/client.properties --add --allow-principal User:***<username>* **--consumer --topic** *<topicname>* **--group** *<consumer group name>* **--resource-pattern-type prefixed**

- Remove the consumer permission from a user:

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<Service IP address of any ZooKeeper node>***:2181/kafka > --remove --allow-principal User:***<Username>* **--consumer --topic** *<Topic name>* **--group** *<Consumer group name>*

  ./kafka-acls.sh --bootstrap-server <*IP address of the Kafkacluster:21007*> **--command-config ../config/client.properties --remove --allow-principal User:***<username>* ***--consumer --topic*** *<topic name>* **--group** *<consumer group name>*

- Delete the consumer permission of a user in batches:

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<Service IP address of any ZooKeeper node>***:2181/kafka > --remove --allow-principal User:***<Username>* **--consumer --topic** *<Topic name>* **--group** *<Consumer group name>* **--resource-pattern-type prefixed**

./kafka-acls.sh --bootstrap-server *<IP address of the Kafkacluster:21007>* **--command-config ../config/client.properties --remove --allow-principal User:***<username>* **--consumer --topic** *<topicname>* **--group** *<consumer group name>* **--resource-pattern-type prefixed**

----End

18.1.2 Creating a Kafka Role

Scenario

Create and configure a Kafka role as an MRS cluster administrator.

This section applies to MRS 3.x or later.

NOTE

- The Kafka role can be created for clusters in security mode, but not for clusters in common mode.
- If the current component uses Ranger for permission control, you need to configure permission management policies based on Ranger. For details, see **Adding a Ranger Access Permission Policy for Kafka**.

Procedure

Step 1 Log in to FusionInsight Manager and choose **System** > **Permission** > **Role**.

Step 2 On the displayed page, click **Create Role** and enter a **Role Name** and **Description**.

Step 3 On the **Configure Resource Permission** page, choose *Name of the desired cluster* > **Kafka**.

Step 4 Select permissions based on service requirements. For details about configuration items, see **Table 18-2**.

Table 18-2 Description

Scenario	Role Authorization
Setting the Kafka administrator permissions	In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Kafka** > **Kafka Manager Privileges**. **NOTE** This permission allows you to create and delete topics, but does not allow you to produce or consume any topics.
Setting the production permission of a user on a topic	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Kafka** > **Kafka Topic Producer And Consumer Privileges**. 2. In the **Permission** column of the specified topic, select **Kafka Producer Permission**.

Scenario	Role Authorization
Setting the consumption permission of a user on a topic	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Kafka** > **Kafka Topic Producer And Consumer Privileges**. 2. In the **Permission** column of the specified topic, select **Kafka Consumer Privileges**.

Step 5 Click **OK**, and return to the **Role** page.

----End

18.1.3 Configuring Token Authentication Information for Kafka Users

Scenario

Operations need to be performed on tokens when the token authentication mechanism is used.

This section applies to Kerberos authentication-enabled clusters of MRS 3.*x* or later.

Prerequisites

- The MRS cluster administrator has understood service requirements and prepared a system user.
- Token authentication has been enabled.
- The Kafka client has been installed.

Procedure

Step 1 Log in as a client installation user to the node on which the Kafka client is installed.

Step 2 Switch to the Kafka client installation directory, for example, **/opt/client**.

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Run the following command to perform user authentication:

kinit *Component service user*

Step 5 Run the following command to switch to the Kafka client installation directory:

cd Kafka/kafka/bin

Step 6 Use **kafka-delegation-tokens.sh** to perform operations on tokens.

- Generate a token for a user.

  ./kafka-delegation-tokens.sh --create --bootstrap-server <*IP1:PORT, IP2:PORT,...***> --max-life-time-period <***Long: max life period in milliseconds***> --command-config <***config file***> --renewer-principal User:<***user name***>**

  Example: **./kafka-delegation-tokens.sh --create --bootstrap-server 192.168.1.1:21007,192.168.1.2:21007,192.168.1.3:21007 --command-config ../config/producer.properties --max-life-time-period -1 --renewer-principal User:username**

- List information about all tokens of a specified user.

  ./kafka-delegation-tokens.sh --describe --bootstrap-server <*IP1:PORT, IP2:PORT,...***> --command-config <***config file***> --owner-principal User:<***user name***>**

  Example: **./kafka-delegation-tokens.sh --describe --bootstrap-server 192.168.1.1:21007,192.168.1.2:21007,192.168.1.3:21007 --command-config ../config/producer.properties --owner-principal User:username**

- Update the token validity period.

  ./kafka-delegation-tokens.sh --renew --bootstrap-server <*IP1:PORT, IP2:PORT,...***> --renew-time-period <***Long: renew time period in milliseconds***> --command-config <***config file***> --hmac <***String: HMAC of the delegation token***>**

  Example: **./kafka-delegation-tokens.sh --renew --bootstrap-server 192.168.1.1:21007,192.168.1.2:21007,192.168.1.3:21007 --renew-time-period -1 --command-config ../config/producer.properties --hmac ABCDEFG**

- Destroy a token.

  ./kafka-delegation-tokens.sh --expire --bootstrap-server <*IP1:PORT, IP2:PORT,...***> --expiry-time-period <***Long: expiry time period in milliseconds***> --command-config <***config file***> --hmac <***String: HMAC of the delegation token***>**

  Example: **./kafka-delegation-tokens.sh --expire --bootstrap-server 192.168.1.1:21007,192.168.1.2:21007,192.168.1.3:21007 --expiry-time-period -1 --command-config ../config/producer.properties --hmac ABCDEFG**

----End

18.2 Using the Kafka Client

Scenario

This section guides users to use a Kafka client in an O&M or service scenario.

This section applies to MRS 3.*x* or later clusters.

Prerequisites

- The client has been installed. For example, the installation directory is **/opt/client**.
- Service component users have been created by the MRS cluster administrator. Machine-machine users need to download the keytab file. A human-machine

user must change the password upon the first login. (Not involved in normal mode)

- After changing the domain name of a cluster, redownload the client to ensure that the **kerberos.domain.name** value in the configuration file of the client is set to the correct server domain name.

Using the Kafka Client

Step 1 Log in to the node where the client is installed as the client installation user.

Step 2 Run the following command to go to the client installation directory:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Run the following command to perform user authentication (skip this step in normal mode):

kinit *Component service user*

Step 5 Run the following command to switch to the Kafka client installation directory:

cd Kafka/kafka/bin

Step 6 Run the following command to use the client tool to view and use the help information:

- **./kafka-console-consumer.sh**: Kafka message reading tool
- **./kafka-console-producer.sh**: Kafka message publishing tool
- **./kafka-topics.sh**: Kafka topic management tool

Step 7 If you need to use kafka-topics.sh to manage Kafka topics, run the following command:

NOTE

- Service IP address of the ZooKeeper node: Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ZooKeeper** > **Instance**, and check and record the service IP address of the ZooKeeper role instance.
- clientPort: You can search for clientPort in all ZooKeeper configuration parameters. The default ports are as follows:

  The default open-source port number is 2181.

  The default customized port number is 24002.

  Port customization/open source: When creating an LTS version cluster, you can set **Component Port** to **Open source** or **Custom**. If **Open source** is selected, the open source port is used. If **Custom** is selected, the customized port is used.
- Kafka cluster IP address: Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka** > **Instance**, and view and record the service IP address of any broker instance.
- The default IP address and port number of the Kafka cluster are 21007 in security mode and 9092 in common mode.
- In MRS 3.3.1 and later versions, Kafka uses **--bootstrap-server** instead of **--zookeeper** to create topics.
- The differences between **--zookeeper** and **--bootstrap-server** are as follows:
  - In **--zookeeper** mode, the client generates a copy allocation scheme. The community supports this mode from the beginning. To reduce the dependency on the ZooKeeper component, the community will delete the support for this mode in later versions. When creating a topic in this mode, you can select a copy allocation policy by combining the **--enable-rack-aware** and **--enable-az-aware** options. Note: The **--enable-az-aware** option can be used only when the cross-AZ feature is enabled on the server, that is, **az.aware.enable** is set to **true**. Otherwise, the execution fails.
  - In **--bootstrap-server** mode, the server generates a copy allocation solution. In later versions, the community supports only this mode for topic management. When a topic is created in this mode, the **--enable-rack-aware** and **--enable-az-aware** options cannot be used to control the copy allocation policy. The **rack.aware.enable** and **az.aware.enable** parameters can be used together to control the copy allocation policy. Note that the **az.aware.enable** parameter cannot be modified; if the cross-AZ feature is enabled during cluster creation, this parameter is automatically set to **true**; the **rack.aware.enable** parameter can be customized.

- Creating a topic:

  ./kafka-topics.sh --create --topic *topic name* **--partitions** *number of partitions occupied by the topic* **--replication-factor** *number of replicas of the topic* **--zookeeper** *Service IP address of the ZooKeeper node:clientPort*/**kafka**

  ./kafka-topics.sh --create --topic *topic name* **--partitions** *number of partitions occupied by the topic* **--replication-factor** *number of replicas of the topic* **--bootstrap-server** *IP address of the Kafka cluster:21007* **--command-config ../config/client.properties**

- List of topics:
  - **./kafka-topics.sh --list --zookeeper** *Service IP address of the ZooKeeper node:clientPort*/**kafka**
  - **./kafka-topics.sh --list --bootstrap-server** *IP address of the Kafka cluster:21007* **--command-config ../config/client.properties**

- Viewing the topic:
  - **./kafka-topics.sh --describe --zookeeper** *Service IP address of the ZooKeeper node:clientPort*/**kafka** --**topic** *topic name*

  - **./kafka-topics.sh --describe --bootstrap-server** *IP address of the Kafka cluster:21007* **--command-config ../config/client.properties --topic** *topic name*

- Modifying a topic:
  - **./kafka-topics.sh --alter --topic** *topic name* **--config** *Configuration item=Configuration item* **--zookeeper** *Service IP address of the ZooKeeper node:clientPort***/kafka**
- Expanding partitions:
  - **./kafka-topics.sh --alter --topic** *topic name* **--zookeeper** *Service IP address of the ZooKeeper node:clientPort***/kafka --command-config Kafka/kafka/config/client.properties --partitions** *Number of partitions after the expansion*
  - **./kafka-topics.sh --alter --topic** *topic name* **--bootstrap-server** *IP address of the Kafka cluster:21007* **--command-config Kafka/kafka/config/client.properties --partitions** *Number of partitions after the expansion*
- Deleting a topic:
  - **./kafka-topics.sh --delete --topic** *topic name* **--zookeeper** *Service IP address of the ZooKeeper node:clientPort***/kafka**
  - **./kafka-topics.sh --delete --topic** *topic name* **--bootstrap-server** *IP address of the Kafka cluster:21007* **--command-config ../config/client.properties**

----End

18.3 Using Kafka to Produce Consumption Data

Scenario

You can use the MRS cluster client to create, query, and delete Kafka topics. You can also log in to the Kafka UI to view the consumption information of the current cluster.

Prerequisites

- The client has been installed in a directory, for example, **/opt/client**. The client directory in the following operations is only an example. Change it based on site requirements.
- If you use KafkaUI to perform operations, you have created a user who has the permission to access the KafkaUI page. If you need to perform operations on the KafkaUI page, for example, create topics, you need to grant related permissions to the user. For details, see **Kafka User Permissions**.

  If you access Manager and KafkaUI for the first time, you need to add a site trust in the browser to continue accessing KafkaUI.

Using the Kafka Client to Produce Consumption Data

Step 1 Install the client. For details, see **Installing a Client**.

Step 2 Access the ZooKeeper instance page.

Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **ZooKeeper** > **Instance**.

Step 3 View the IP addresses of the ZooKeeper role instance.

Record any IP address of the ZooKeeper instance.

Step 4 Log in to the node where the client is installed.

Step 5 Run the following command to switch to the client installation directory, for example, **/opt/client/Kafka/kafka/bin**.

cd /opt/client/Kafka/kafka/bin

Step 6 Run the following command to configure environment variables:

source /opt/client/bigdata_env

Step 7 If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the current user. If Kerberos authentication is disabled for the current cluster, skip this step.

kinit *Kafka user*

Step 8 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ZooKeeper**, and click the **Configurations** tab and then **All Configurations**. On the displayed page, search for the **clientPort** parameter and record its value.

Step 9 Create a topic.

sh kafka-topics.sh --create --topic *Topic name* **--partitions** *Number of partitions occupied by the topic* **--replication-factor** *Number of replicas of the topic* **--zookeeper** *IP address of the node where the ZooKeeper instance resides:clientPort*/**kafka**

Example: **sh kafka-topics.sh --create --topic TopicTest --partitions 3 --replication-factor 3 --zookeeper 10.10.10.100:2181/kafka**

NOTE

You can search for **clientPort** in all ZooKeeper configuration parameters to obtain the value of **clientPort**. The default ports are as follows:

- The default open-source port number is **2181**.
- The default customized port number is **24002**.

Port customization/open source: When creating an LTS version cluster, you can set **Component Port** to **Open source** or **Custom**. If **Open source** is selected, the open source port is used. If **Custom** is selected, the customized port is used.

Step 10 Run the following command to view the topic information in the cluster:

sh kafka-topics.sh --list --bootstrap-server *IP address of the Kafka cluster:21007* **--command-config ../config/client.properties**

Example: **sh kafka-topics.sh --list --bootstrap-server** 10.10.10.100*:21007* **--command-config ../config/client.properties**

Step 11 Delete the topic created in **Step 9**.

sh kafka-topics.sh --delete --topic *Topic name* **--zookeeper** *IP address of the node where the ZooKeeper instance resides*:*clientPort*/**kafka**

Example: **sh kafka-topics.sh --delete --topic TopicTest --zookeeper 10.10.10.100:2181/kafka**

----End

Using KafkaUI to View Consumption Information

Step 1 Access the Kafka UI.

1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

   If you need to perform related operations on the page, for example, creating a topic, you need to grant related permissions to the user. For details, see **Kafka User Permissions**.

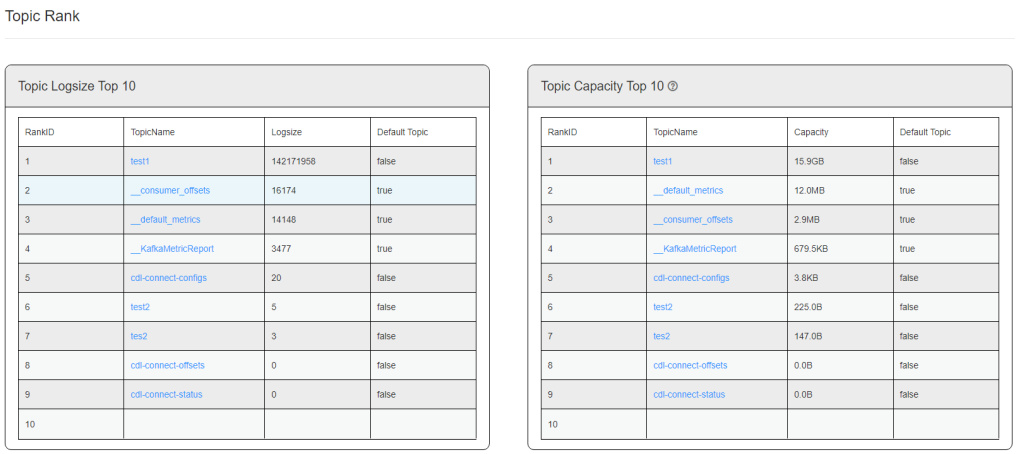
2. On the right of **KafkaManager WebUI**, click the URL to access Kafka UI.

Step 2 In the **Cluster Summary** area, view the number of existing topics, brokers, and consumer groups in the current cluster.

Step 3 You can click the number under **Brokers**, **Topics**, or **Consumer Group** to go to the corresponding page and view and perform operations on the information.

Step 4 In the **Cluster Action** area, you can create topics and migrate partitions. For details, see sections **Creating a Kafka Topic** and **Migrating Data Between Kafka Nodes**.

Step 5 In the **Topic Rank** column, view top 10 topics by the number of topic logs, data volume, incoming data volume, and outgoing data volume in the current cluster.


Topic Rank

Topic Logsize Top 10

RankID	TopicName	Logsize	Default Topic
1	test1	142171958	false
2	__consumer_offsets	16174	true
3	__default_metrics	14148	true
4	__KafkaMetricReport	3477	true
5	cdl-connect-configs	20	false
6	test2	5	false
7	tes2	3	false
8	cdl-connect-offsets	0	false
9	cdl-connect-status	0	false
10			

Topic Capacity Top 10

RankID	TopicName	Capacity	Default Topic
1	test1	15.9GB	false
2	__default_metrics	12.0MB	true
3	__consumer_offsets	2.9MB	true
4	__KafkaMetricReport	679.5KB	true
5	cdl-connect-configs	3.8KB	false
6	test2	225.0B	false
7	tes2	147.0B	false
8	cdl-connect-offsets	0.0B	false
9	cdl-connect-status	0.0B	false
10			

Step 6 Click a topic name in the **TopicName** column to go to the topic details page. For details about operations on the page, see **Viewing Kafka Data Production and Consumption Details**.

----End

18.4 Creating a Kafka Topic

Scenario

You can use the cluster client or KafkaUI to create Kafka topics based on service requirements. Management permission is required for clusters with Kerberos authentication enabled.

Prerequisites

You have installed the Kafka client.

Creating a Kafka Topic Using the Kafka Client

Step 1 Access the ZooKeeper instance page.

Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **ZooKeeper** > **Instance**.

Step 2 View the IP addresses of the ZooKeeper role instance.

Record any IP address of the ZooKeeper instance.

Step 3 Prepare the client based on service requirements. Log in to the node where the client is installed.For details, see **Using an MRS Client**.

Step 4 Run the following command to switch to the client installation directory, for example, **/opt/client/Kafka/kafka/bin**.

cd /opt/client/Kafka/kafka/bin

Step 5 Run the following command to configure environment variables:

source /opt/client/bigdata_env

Step 6 Run the following command to perform user authentication (skip this step in normal mode):

kinit *Component service user*

Step 7 Use **kafka-topics.sh** to create Kafka topics.

./kafka-topics.sh --create --topic *topic name* **--partitions** *number of partitions occupied by the topic***--replication-factor** *number of replicas of the topic***--zookeeper** *IP address of any ZooKeeper node:clientPort***/kafka**

./kafka-topics.sh --create --topic *topic name* **--partitions** *number of partitions occupied by the topic***--replication-factor** *number of replicas of the topic* **--bootstrap-server** *IP address of the Kafkacluster:21007* **--command-config ../config/client.properties**

----End

Creating a Kafka Topic Using KafkaUI

Step 1 Access the Kafka UI.

1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

   If you need to perform related operations on the page, for example, creating a topic, you need to grant related permissions to the user. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access Kafka UI.

Step 2 Click **Create Topic**. On the page that is displayed, set parameters based on **Table 18-3** and click Create.

Table 18-3 Topic information

Parameter	Description	Remarks
Topic	Topic name, which can contain a maximum of 249 characters. Only letters, digits, hyphens (-), and underscores (_) are allowed.	Example: **kafka_ui**
Partitions	Number of topic partitions. The value must be greater than or equal to 1. The default value is **3**.	-
Replication Factor	Replication factor of a topic. The value ranges from 1 to *N*. *N* indicates the number of brokers in the current cluster. The default value is **2**.	-

NOTE

- You can click **Advanced Options** to set advanced topic parameters based on service requirements. Generally, retain the default values.
- In a cluster in security mode, the user who creates a topic must belong to the **kafkaadmin** user group. Otherwise, the topic cannot be created due to authentication failure.
- In a cluster in non-security mode, no authentication is required for creating a topic. That is, any user can create a topic.

----End

18.5 Accessing Messages in Kafka Topics

Scenario

You can use the Kafka client or Kafka UI to view the current consumption information based on service requirements.

This section applies to MRS 3.*x* or later.

Prerequisites

When using the Kafka client, ensure that the following conditions are met:

- The MRS cluster administrator has understood service requirements and prepared a system user.
- The Kafka client has been installed.

Using the Kafka Client to View the Consumption Information

Step 1 Log in as a client installation user to the node on which the Kafka client is installed.

Step 2 Switch to the Kafka client installation directory, for example, **/opt/client**.

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Run the following command to perform user authentication (skip this step in normal mode):

kinit *Component service user*

Step 5 Run the following command to switch to the Kafka client installation directory:

cd Kafka/kafka/bin

Step 6 Run the **kafka-consumer-groups.sh** command to check the current consumption status.

- Check the Consumer Group list on Kafka saved by Offset:

./kafka-consumer-groups.sh --list --bootstrap-server <*Service IP address of any node where a broker instances is located*:*Port number of the Kafka cluster*> **--command-config ../config/consumer.properties**

Example: **./kafka-consumer-groups.sh --bootstrap-server 192.168.1.1:21007 --list --command-config ../config/consumer.properties**

- Check the consumption status of Consumer Group on Kafka saved by Offset:

  ./kafka-consumer-groups.sh --describe --bootstrap-server <*Service IP address of any node where a broker instances is located*:*Port number of the Kafka cluster*> **--group** *Consumer group name* **--command-config ../config/consumer.properties**

  Example: **./kafka-consumer-groups.sh --describe --bootstrap-server 192.168.1.1:21007 --group example-group --command-config ../config/consumer.properties**

NOTICE

- Ensure that the current consumer is online and consumes data.
- Configure the **group.id** in the **consumer.properties** configuration file and **--group** in the command to the group to be queried.
- The Kafka cluster's IP port number is 21007 in security mode and 9092 in normal mode.

----End

Using KafkaUI to View the Current Consumption Status

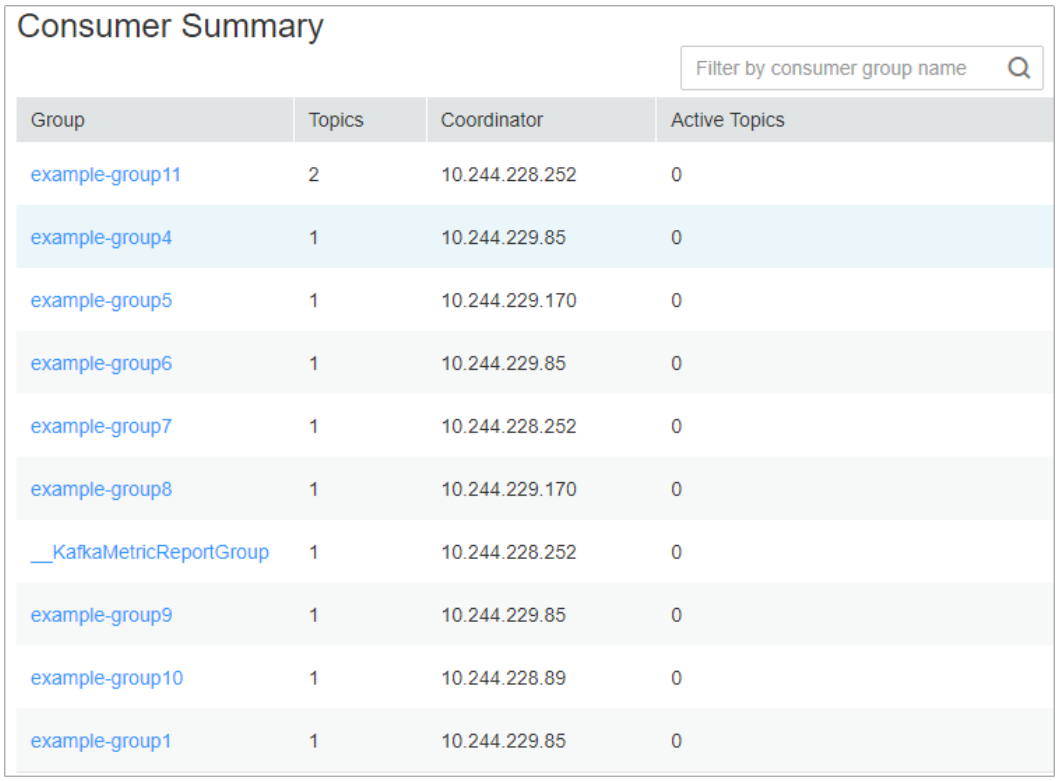
Step 1 Access the Kafka UI.

1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

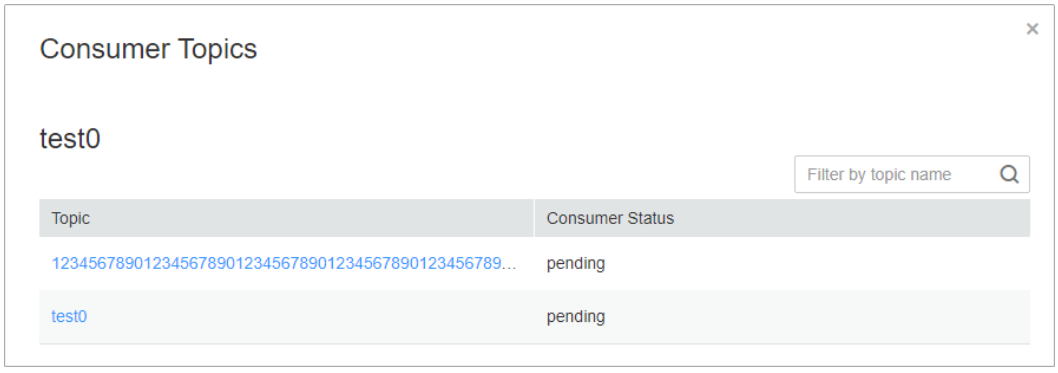
   If you need to perform related operations on the page, for example, creating a topic, you need to grant related permissions to the user. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access Kafka UI.

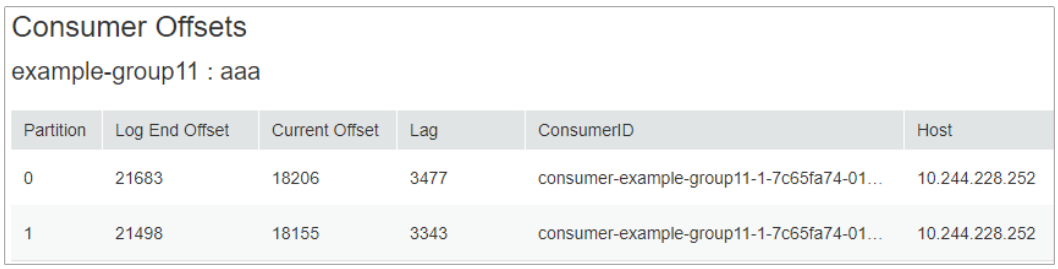
Step 2 Click **Consumers**. On the consumer group details page that is displayed, you can view all consumer groups in the current cluster and the IP address of the node where each consumer group coordinator is located. In the upper right corner of the page, you can enter a consumer group name to search for the specified consumer group.



Step 3 In the **Consumer Summary** area, you can view the existing consumer groups in the current cluster. You can click a consumer group name to view the topics consumed by the consumer group. Consumed topics can be in the **pending** or **running** state. **pending** indicates that the topic has been consumed but not being consumed. **running** indicates that the topic is being consumed. You can enter a topic name in the upper right corner of the dialog box to filter topics.

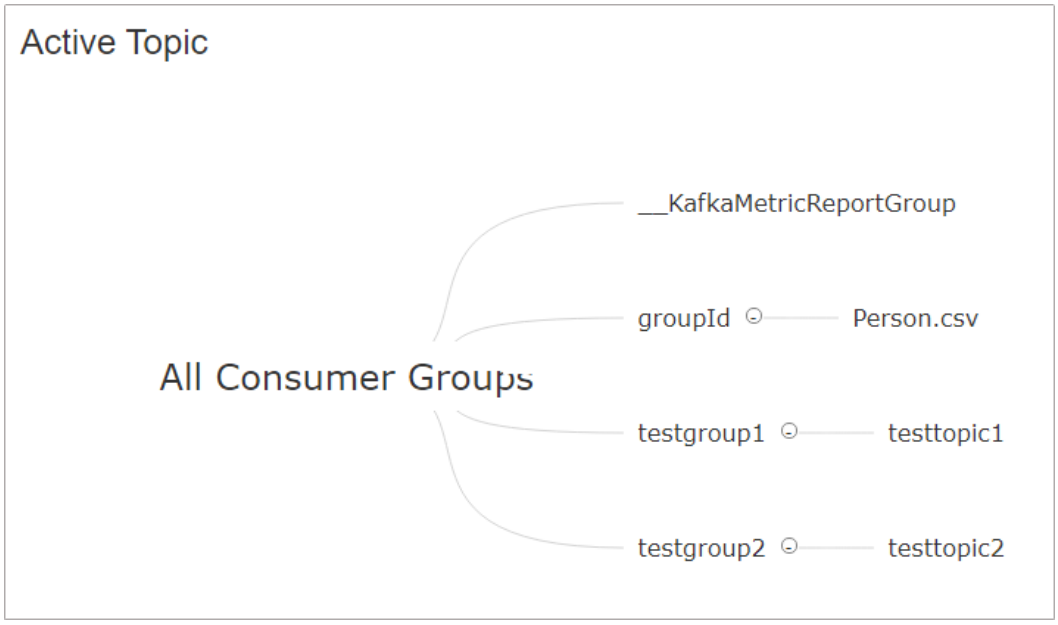


Step 4 Click a topic name. On the **Consumer Offsets** page that is displayed, view the topic consumption details.



Step 5 View the consumption lineage chart.

Click **Consumers**. The consumer group details page is displayed. In the **Active Topic** area, view all consumer groups in the current cluster and topics that are being consumed by each consumer group.



NOTE

MRS clusters do not support redirection by clicking a consumer group name.

----End

18.6 Managing Kafka Topics

18.6.1 Viewing Kafka Topic Information

Scenario

You can view information about created Kafka topics on the Manager or Kafka UI.

Viewing Kafka Topic Information on the Manager

Step 1 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **Kafka**.

Step 2 Click **KafkaTopicMonitor**.

All topics are displayed in the list by default. You can view the number of partitions and replicas of the topics.

Step 3 Click the desired topic in the list to view its details.

NOTE

If the following operations are performed, Kafka topic monitoring may not be displayed:

- Capacity expansion or reduction has been performed on Kafka or ZooKeeper.
- Instances have been added to or deleted from Kafka or ZooKeeper.
- The Elasticsearch service is reinstalled.
- Kafka is switched to another ZooKeeper service.

Perform the following steps to rectify the fault:

1. Log in to the active OMS node of the cluster and run the following command to switch to user **omm**:

   su - omm

2. Restart the CEP service.

   restart_app cep

   Wait for 3 minutes and check the Kafka topic monitoring again.

----End

Viewing Kafka Topic Information on KafkaUI

Step 1 Access the Kafka UI.

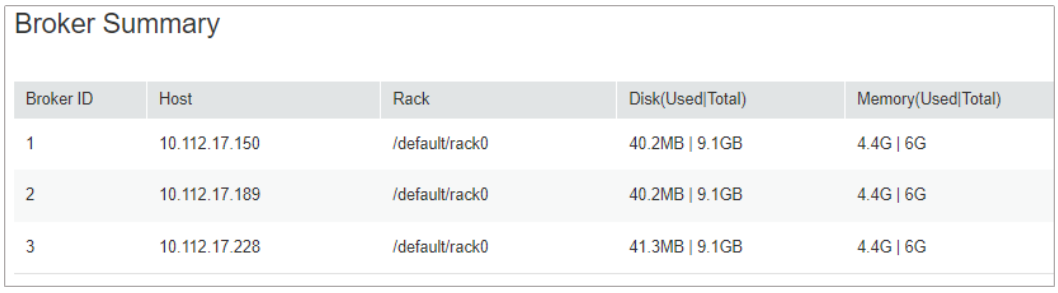
1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

   If you need to perform related operations on the page, for example, creating a topic, you need to grant related permissions to the user. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access Kafka UI.

Step 2 Click **Brokers**. The broker details page is displayed.

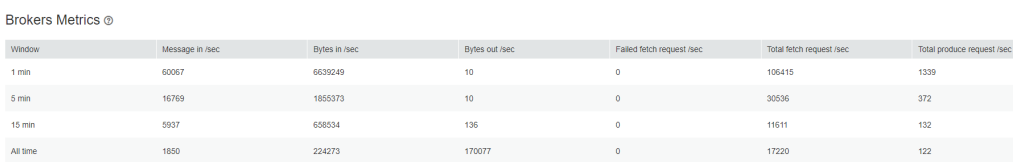
Step 3 In the **Broker Summary** area, you can view **Broker ID**, **Host**, **Rack**, **Disk(Used|Total)**, and **Memory(Used|Total)** of brokers.



Broker Summary

Broker ID	Host	Rack	Disk(Used|Total)	Memory(Used|Total)
1	10.112.17.150	/default/rack0	40.2MB | 9.1GB	4.4G | 6G
2	10.112.17.189	/default/rack0	40.2MB | 9.1GB	4.4G | 6G
3	10.112.17.228	/default/rack0	41.3MB | 9.1GB	4.4G | 6G

Step 4 In the **Brokers Metrics** area, you can view the JMX metrics of the broker node data traffic, including the average number of incoming messages per second, number of bytes of incoming messages per second, number of bytes of outgoing messages per second, and number of failed requests per second, total number of requests per second, and number of production requests per second in different time windows.



Brokers Metrics

Window	Message in /sec	Bytes in /sec	Bytes out /sec	Failed fetch request /sec	Total fetch request /sec	Total produce request /sec
1 min	60067	6639249	10	0	106415	1339
5 min	16769	1855373	10	0	30536	372
15 min	5937	658534	136	0	11611	132
All time	1850	224273	170077	0	17220	122

Step 5 In the upper right corner of the page, you can enter a host IP address or rack configuration information to search for a broker.

----End

18.6.2 Modifying Kafka Topic Configurations

Scenario

You can use a cluster client to create Kafka topics based on service requirements. Management permission is required for clusters with Kerberos authentication enabled. You can also modify topic configurations on Kafka UI.

NOTE

- In security mode, when modifying topic configurations on the Kafka UI, ensure that the Kafka UI login user belongs to the **kafkaadmin** user group or the corresponding operation permission is granted to the user. Otherwise, the authentication fails.
- In non-security mode, Kafka UI does not authenticate any operation.

Modifying Kafka Topics Using the Kafka Client

Step 1 Access the ZooKeeper instance page.

Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **ZooKeeper** > **Instances**.

Step 2 View the IP addresses of the ZooKeeper role instance.

Record any IP address of the ZooKeeper instance.

Step 3 Prepare a client based on service requirements. Log in to the node where the client is installed by referring to **Using an MRS Client**.

Step 4 Run the following command to switch to the client directory, for example, /opt/client /Kafka/kafka/bin:

cd /opt/client/Kafka/kafka/bin

Step 5 Run the following command to set environment variables:

source /opt/client/bigdata_env

Step 6 Run the following command to perform user authentication (skip this step in normal mode).

kinit *Component service user*

Step 7 Use **kafka-topics.sh** to modify the topic.

./kafka-topics.sh --alter --topic *topic name***--config** *configuration item=configuration value* **--zookeeper** *service IP address of any ZooKeeper node:clientPort***/kafka**

Step 8 Use **kafka-topics.sh** to view the modified topic.

- **./kafka-topics.sh --describe --zookeeper** *service IP address of any ZooKeeper node:clientPort***/kafka** --**topic** *topic name*

- **./kafka-topics.sh --describe --bootstrap-server** *IP address of the Kafkacluster:21007* **--command-config ../config/client.properties --topic** *topic name*

----End

Modifying Kafka Topics Using KafkaUI

Step 1 Access the Kafka UI.

1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

   If you need to perform related operations on the page, for example, creating a topic, you need to grant related permissions to the user. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access Kafka UI.

Step 2 Click **Topics**. The topic management page is displayed.

Step 3 Click Action > Config in the Operation column of the item to be modified. On the page that is displayed, modify the Key and Value of the topic. To add multiple values, click  . Click **OK**.

----End

18.6.3 Adding Kafka Topic Partitions

Scenario

You can add Kafka topic partitions on the Kafka UI.

NOTE

- In security mode, the user who migrates a partition must belong to the **kafkaadmin** user group. Otherwise, the operation fails due to authentication failure.
- In non-security mode, Kafka UI does not authenticate any operation.

Adding a Partition

Step 1 Access the Kafka UI.

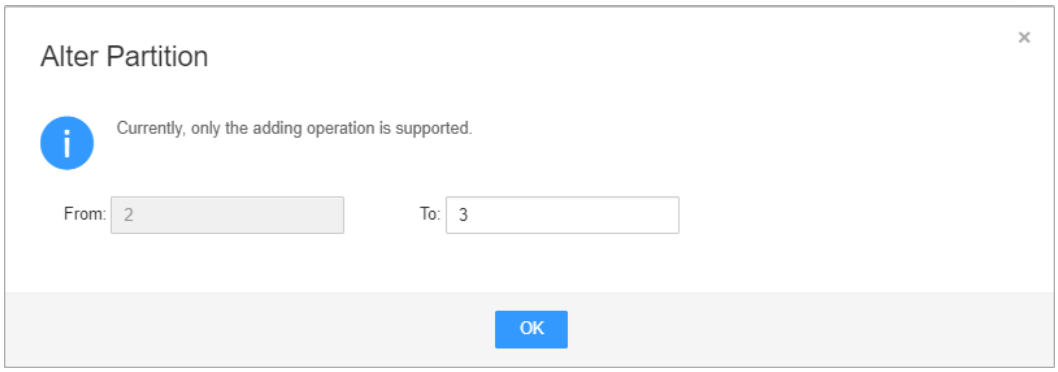
1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

   If you need to perform related operations on the page, for example, creating a topic, you need to grant related permissions to the user. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access Kafka UI.

Step 2 Click **Topics**. The topic management page is displayed.

Step 3 In the **Operation** column of the item to be modified, choose **Action** > **Alter**. On the displayed page, modify the topic partition.



NOTE

Currently, you can only add partitions to a cluster. That is, the number of partitions after modification must be greater than the number of original partitions.

Step 4 Click **OK**.

----End

18.6.4 Managing Messages in Kafka Topics

Scenario

You can produce or consume messages in Kafka topics using the MRS cluster client.

Prerequisites

- The cluster client has been installed.
- For clusters with Kerberos authentication enabled, you need to create a service user on Manager in advance. The user has the permission to perform operations in Kafka topics.

Managing Messages

Step 1 Go to the Kafka service page.

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka**.

Step 2 Click **instance**. Query the IP addresses of the Kafka broker instances.

Record the IP address of any Kafka instance.

Step 3 Prepare the client based on service requirements. Log in to the node where the client is installed. For details, see **Using an MRS Client** .

Step 4 Run the following command to switch to the client installation directory, for example, **/opt/client/Kafka/kafka/bin**.

cd /opt/client/Kafka/kafka/bin

Step 5 Run the following command to configure environment variables:

source /opt/client/bigdata_env

Step 6 For clusters with Kerberos authentication enabled, run the following command to authenticate the user. For clusters with Kerberos authentication disabled, skip this step.

kinit *Kafka user*

Step 7 Manage messages in Kafka topics using the following commands:

- Producing messages

  sh kafka-console-producer.sh --broker-list *IP address of the node where the broker instance is located*:**9092 --topic** *Topic name* **--producer.config /opt/client/Kafka/kafka/config/producer.properties**

  A topic must be created in advance. You can input specified information as the messages produced by the producer and then press **Enter** to send the messages. To end message producing, press **Ctrl + C** to exit.

- Consuming messages

  Start another client connection and run the following commands to consume messages in the topic:

  cd /opt/client/Kafka/kafka/bin

  source /opt/client/bigdata_env

  sh kafka-console-consumer.sh --topic *Topic name* **--bootstrap-server** *IP address of the node where the broker instance is located*:**9092 --consumer.config /opt/client/Kafka/kafka/config/consumer.properties**

  In the configuration file, **group.id** (indicating the consumer group) is set to **example-group1** by default. Users can change the value as required. The value takes effect each time consumption occurs.

  By default, the system reads unprocessed messages in the current consumer group when the command is executed. If a new consumer group is specified in the configuration file and the **--from-beginning** parameter is added to the command, the system reads all messages that have not been automatically deleted in Kafka.

NOTE

- For the IP address of the node where the Kafka instance locates, use the IP address of any broker instance.
- If Kerberos authentication is enabled, change the port to **21007**.
- By default, the ZooKeeper's **clientPort** value is **2181**.

----End

18.6.5 Viewing Kafka Data Production and Consumption Details

Scenario

On Kafka UI, you can view topic details, modify topic configurations, add topic partitions, delete topics, and view the number of data records produced in different time segments in real time.

NOTE

- In security mode, Kafka UI does not authenticate the operation of viewing topic details. That is, any user can query topic information. To modify topic configurations, add topic partitions, or delete topics, ensure that the Kafka UI login user belongs to the **kafkaadmin** user group or grant the corresponding operation permissions to the user. Otherwise, the authentication fails.
- In non-security mode, Kafka UI does not authenticate any operation.

Viewing Production and Consumption Details

Step 1 Access the Kafka UI.

1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

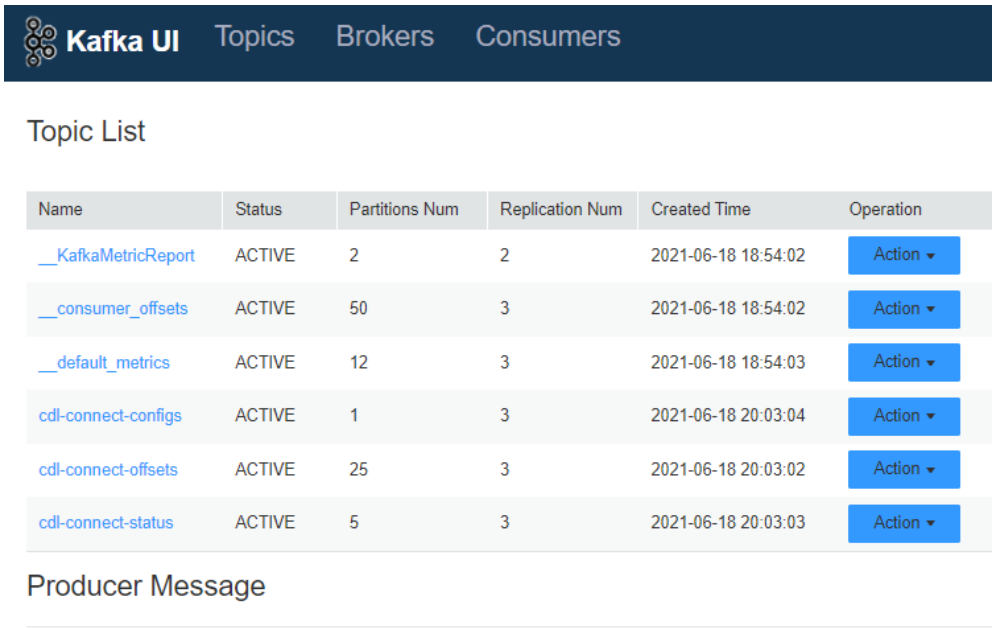
   If you need to perform related operations on the page, for example, creating a topic, you need to grant related permissions to the user. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access Kafka UI.

Step 2 Click **Topics**. The topic management page is displayed.

You can perform the following operations:

- In the **Topic List** area, you can view the names, status, number of partitions, creation time, and number of replicas of topics created in the current cluster.

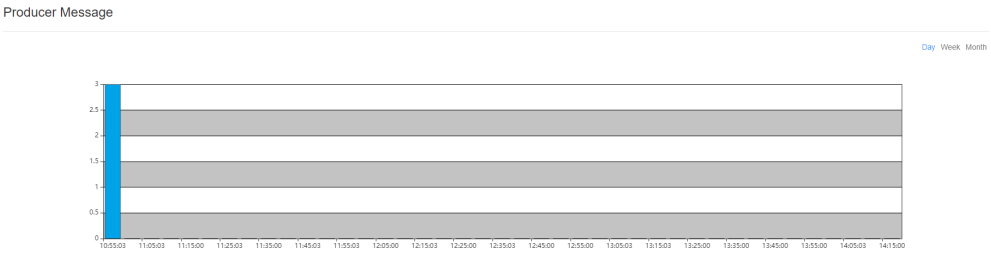


- Click a topic name to go to the topic details page. On this page, you can view details about topics and partitions.

Partition Summary

Partition Id	Leader	Replicas	In Sync Replicas	Logsize	Start Offset	End Offset
0	1	[1, 2, 3]	[1,2,3]	0.0B	0	0
1	2	[2, 3, 1]	[2,3,1]	0.0B	0	0
2	3	[3, 1, 2]	[3,1,2]	0.0B	0	0
3	1	[1, 3, 2]	[1,3,2]	0.0B	0	0
4	2	[2, 1, 3]	[2,1,3]	0.0B	0	0
5	3	[3, 2, 1]	[3,2,1]	3.0MB	0	14583
6	1	[1, 2, 3]	[1,2,3]	0.0B	0	0
7	2	[2, 3, 1]	[2,3,1]	0.0B	0	0
8	3	[3, 1, 2]	[3,1,2]	0.0B	0	0
9	1	[1, 3, 2]	[1,3,2]	0.0B	0	0

- In the **Producer Message** area, you can select **Day**, **Week**, or **Month** based on service requirements to view the number of data records produced in the topic.



- Modify topic configurations.

  Click **Action** > **Config** in the **Operation** column of the item to be modified. On the page that is displayed, modify the Key and Value of the topic. To add multiple values, click +. Click **OK**.

- Search for a topic.

  In the upper right corner of the page, enter a topic name to search for the topic.

- Delete a topic.

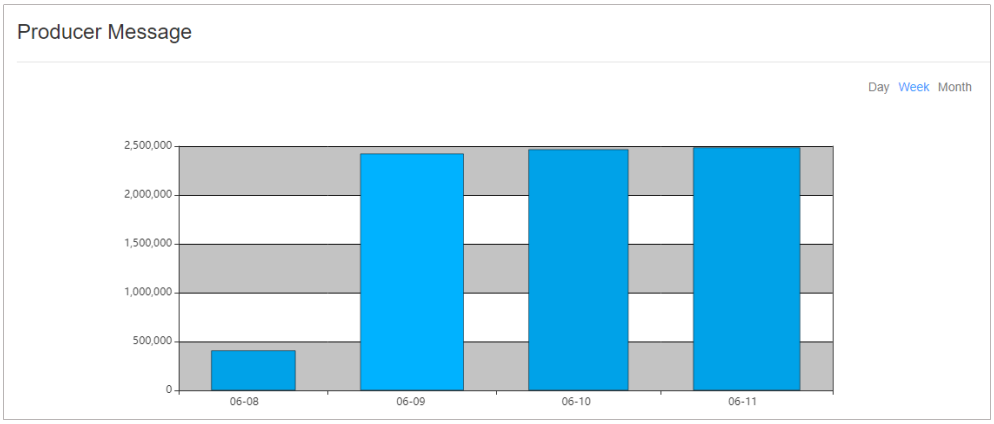
  In the **Operation** column of the item to be modified, choose **Action** > **Delete**. In the confirmation dialog box that is displayed, click **OK**.

  NOTE

  The default built-in topics cannot be deleted.

- Check the number of production data records.

  In the **Producer Message** area, you can select **Day**, **Week**, or **Month** to view the number of data records produced in different time segments in the current cluster.



----End

18.7 Enterprise-Class Enhancements of Kafka

18.7.1 Configuring Kafka HA and High Reliability

Scenario

For the Kafka message transmission assurance mechanism, different parameters are available for meeting different performance and reliability requirements. This section describes how to configure Kafka high availability (HA) and high reliability parameters.

This section applies to MRS 3.x or later.

Impact on the System

- Impact of HA and high performance configurations:

  After HA and high performance are configured, the data reliability decreases. Specifically, data may be lost of disks or nodes are faulty.

- Impact of high reliability configurations:
  - Deteriorated performance

    If **ack** is set to **–1**, data written is considered as successful only when data is written to multiple replicas. As a result, the delay of a single message increases and the client processing capability decreases. The impact is subject to the actual test data.

  - Reduced availability

    A replica that is not in the ISR list cannot be elected as a leader. If the leader goes offline and other replicas are not in the ISR list, the partition remains unavailable until the leader node recovers. When the node where a replica of a partition is located is faulty, the minimum number of successful replicas cannot be met. As a result, service writing fails.

- If parameters are at the service level, Kafka needs to be restarted. You are advised to modify the service-level configuration in the change window.

Parameter Description

- If services require high availability and high performance,

  Go to the Kafka service configuration page by referring to **Modifying Cluster Service Configuration Parameters** and set the parameters in **Table 18-4** on the server.

  Table 18-4 Server HA and high performance parameters

Parameter	Default Value	Description
unclean.leader.election.enable	true	Specifies whether a replica that is not in the ISR can be selected as the leader. If this parameter is set to **true**, data may be lost.
auto.leader.rebalance.enable	true	Specifies whether the leader automated balancing function is used. If this parameter is set to **true**, the controller periodically balances the leader of each partition on all nodes and assigns the leader to a replica with a higher priority.
min.insync.replicas	1	Specifies the minimum number of replicas to which data is written when **acks** is set to **-1** for the Producer.

  Set the parameters listed in **Table 18-5** in the client configuration file **producer.properties**. The path for storing **producer.properties** is *Kafka client installation path* **/Kafka/kafka/config/producer.properties**.

Table 18-5 Client HA and high performance parameters

Parameter	Default Value	Description
acks	1	The leader needs to check whether the message has been received and determine whether the required operation has been processed. This parameter affects message reliability and performance. • If this parameter is set to **0**, the producer does not wait for any response from the server, and the message is considered successful. • If this parameter is set to **1**, when the leader of the replica verifies that data has been written into the cluster, the leader returns a response without waiting for data to be written to all replicas. In this case, if the leader is abnormal when the leader makes the confirmation but replica synchronization is not complete, data will be lost. • If this parameter is set to **-1**, the message is considered to be successfully received only when all synchronized replicas are confirmed. If the **min.insync.replicas** parameter is also configured, data can be written into multiple replicas. In this case, records will not be lost as long as one replica remains active.

- To ensure high data reliability for services,

  set the parameters listed in **Table 18-6** on the server. For details about the parameter configuration entry, see **Modifying Cluster Service Configuration Parameters**.

  Table 18-6 Server HA parameters

Parameter	Recommended Value	Description
unclean.leader.election.enable	false	A replica that is not in the ISR list cannot be elected as a leader.
min.insync.replicas	2	Specifies the minimum number of replicas to which data is written when **acks** is set to **-1** for the Producer. Ensure that the value of **min.insync.replicas** is equal to or less than that of **replication.factor**.

Set the parameters listed in **Table 18-7** in the client configuration file **producer.properties**. The path for storing **producer.properties** is *Kafka client installation path* **/Kafka/kafka/config/producer.properties**.

Table 18-7 Server HA parameters

Parameter	Recommended Value	Description
acks	-1	The leader needs to check whether the message has been received and determine whether the required operation has been processed. If this parameter is set to **-1**, the message is considered to be successfully received only when all replicas in the ISR list have confirmed to receive the message. This parameter is used along with **min.insync.replicas** to ensure that multiple copies are successfully written. As long as one copy is active, the record will not be lost. If this parameter is set to **-1**, the production performance deteriorates. Therefore, you need to set this parameter based on the actual situation.

Configuration Suggestions

Configure parameters based on requirements on reliability and performance in the following service scenarios:

- For valued data, you are advised to configure RAID1 or RAID5 for Kafka data directory disks to improve data reliability when a single disk is faulty.
- For parameters that can be modified at the topic level, the service level configurations are used by default.

  These parameters can be separately configured based on topic reliability requirements. For example, log in to the Kafka client as user **root**, and run the following command to configure the reliability parameter with topic named test in the client installation directory:

  cd Kafka/kafka/bin

  kafka-configs.sh --bootstrap-server *192.168.1.205*:**21007 --command-config ../config/client.properties --alter --topic test --add-config unclean.leader.election.enable=false --config min.insync.replicas=2**
- If parameters are at the service level, Kafka needs to be restarted. You are advised to modify the service-level configuration in the change window.

18.7.2 Configuring a Secure Transmission Protocol for Kafka Data

This section applies to MRS 3.*x* or later.

Brief Introduction to Kafka APIs

- Producer API

  Indicates the API defined in **org.apache.kafka.clients.producer.KafkaProducer**. When **kafka-console-producer.sh** is used, the API is used by default.

- Consumer API

  Indicates the API defined in **org.apache.kafka.clients.consumer.KafkaConsumer**. When **kafka-console-consumer.sh** is used, the API is used by default.

NOTE

In MRS 3.*x* or later, Kafka no longer support old Producer or Consumer APIs.

Protocol Description for Accessing Kafka

The protocols used to access Kafka are as follows: PLAINTEXT, SSL, SASL_PLAINTEXT, and SASL_SSL.

When Kafka service is started, the listeners using the PLAINTEXT and SASL_PLAINTEXT protocols are started. You can set **ssl.mode.enable** to **true** in Kafka service configuration to start listeners using SSL and SASL_SSL protocols. The following table describes the four protocols:

For details about how to view or configure parameters, see **Modifying Cluster Service Configuration Parameters**.

Protocol	Description	Default Port
PLAINTEXT	Supports plaintext access without authentication.	Obtain the value of **port**. The default value is 9092
SASL_PLAINTEXT	Supports plaintext access with Kerberos authentication.	Obtain the value of **sasl.port**. The default value is 21007
SSL	Supports SSL-encrypted access without authentication.	Obtain the value of **ssl.port**. The default value is 9093
SASL_SSL	Supports SSL-encrypted access with Kerberos authentication.	Obtain the value of **sasl-ssl.port**. The default value is 21009

ACL Settings for a Topic

To view and set topic permission information, run the **kafka-acls.sh** script on the Linux client. For details, see **Kafka User Permissions**.

Use of Kafka APIs in Different Scenarios

- Scenario 1: accessing the topic with an ACL

Used API	User Group	Client Parameter	Server Parameter	Accessed Port
API	Users need to meet one of the following conditions: • Assigned the **System_administrator** role • In the **kafkaadmin** group • In the **kafkasuperuser** group • In the **kafka** group and be authorized	security.inter.broker.protocol=SASL_PLAINTEXT sasl.kerberos.service.name = kafka	-	sasl.port (The default number is 21007.)
		security.protocol=SASL_SSL sasl.kerberos.service.name = kafka	Set **ssl.mode.enable** to **true**.	sasl-ssl.port (The default number is 21009.)

- Scenario 2: accessing the topic without an ACL

Used API	User Group	Client Parameter	Server Parameter	Accessed Port
API	Users need to meet one of the following conditions: • Assigned the **System_administrator** role • In the **kafkaadmin** group • In the **kafkasuperuser** group	security.protocol=SASL_PLAINTEXT sasl.kerberos.service.name = kafka	-	sasl.port (The default number is 21007.)

Used API	User Group	Client Parameter	Server Parameter	Accessed Port
	Users are in the **kafka** group.		Set **allow.everyone.if.no.acl.found** to **true**. **NOTE** In normal mode, the server parameter **allow.everyone.if.no.acl.found** does not need to be modified.	sasl.port (The default number is 21007.)
	Users need to meet one of the following conditions: ● Assigned the **System_administrator** role ● In the **kafkaadmin** group ● In the **kafkasuperuser** group	security.protocol=SASL_SSL sasl.kerberos.service.name = kafka	Set **ssl.mode.enable** to **true**.	sasl-ssl.port (The default number is 21009.)
	Users are in the **kafka** group.		1. Set **allow.everyone.if.no.acl.found** to **true**. 2. Set **ssl.mode.enable** to **true**.	sasl-ssl.port (The default number is 21009.)
	-	security.protocol=PLAINTEXT	Set **allow.everyone.if.no.acl.found** to **true**.	port (The default number is 9092.)

Used API	User Group	Client Parameter	Server Parameter	Accessed Port
	-	security.protocol=SSL	1. Set **allow.everyone.if.no.acl.found** to **true**. 2. Set **ssl.mode.enable** to **true**.	ssl.port (The default number is 9063.)

18.7.3 Configuring the Kafka Data Balancing Tool

Scenario

This section describes how to use the Kafka balancing tool on a client to balance the load of the Kafka cluster based on service requirements in scenarios such as node decommissioning, node recommissioning, and load balancing.

Prerequisites

- The MRS cluster administrator has understood service requirements and prepared a Kafka administrator (belonging to the **kafkaadmin** group. It is not required for the normal mode.).
- The Kafka client has been installed.

Procedure

Step 1 Log in as a client installation user to the node on which the Kafka client is installed.

Step 2 Switch to the Kafka client installation directory, for example, **/opt/client**.

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Run the following command to authenticate the user (skip this step in normal mode):

kinit *Component service user*

Step 5 Run the following command to switch to the Kafka client installation directory:

cd Kafka/kafka

Step 6 Run the **kafka-balancer.sh** command to balance user cluster. The commonly used commands are:

- Run the **--run** command to perform cluster balancing:

./bin/kafka-balancer.sh --run --zookeeper *<ZooKeeper service IP address of any ZooKeeper node:zkPort/**kafka**>* **--bootstrap-server** *<Kafka cluster IP: port>* **--throttle 10000000 --consumer-config config/consumer.properties --show-details**

This command consists of generation and execution of the balancing solution. **--show-details** is optional, indicating whether to print the solution details. **--throttle** indicates the bandwidth limit during the execution of the balancing solution. The unit is bytes per second (bytes/sec).

- Run the **--run** command to decommission a node:

  ./bin/kafka-balancer.sh --run --zookeeper *<Service IP address of any ZooKeeper node:zkPort/**kafka**>* **--bootstrap-server** *<Kafka cluster IP address: port>* **--throttle 10000000 --consumer-config config/consumer.properties --remove-brokers** *<BrokerId list>* **--force**

  In the command, **--remove-brokers** indicates the list of broker IDs to be deleted. Multiple broker IDs are separated by commas (,). **--force** is optional, indicating that the disk usage alarm is ignored and the migration solution is forcibly generated.

  NOTE

  This command migrates data on the Broker nodes to be decommissioned to other Broker nodes.

- Run the following command to view the execution status:

  ./bin/kafka-balancer.sh --status --zookeeper *<Service IP address of any ZooKeeper node:zkPort/**kafka**>*

- Run the following command to generate a balancing solution:

  ./bin/kafka-balancer.sh --generate --zookeeper *<Service IP address of any ZooKeeper node:zkPort/**kafka**>* **--bootstrap-server** *<Kafka cluster IP address:port>* **--consumer-config config/consumer.properties**

  This command is used to generate a migration solution based on the current cluster status and print the solution to the console.

- Clearing the intermediate status

  ./bin/kafka-balancer.sh --clean --zookeeper *<Service IP address of any ZooKeeper node:zkPort/**kafka**>*

  This command is used to clear the intermediate status information on the ZooKeeper when the migration is not complete.

NOTICE

The port number of the Kafka cluster's IP address is 21007 in security mode and 9092 in normal mode.

----End

Troubleshooting

During partition migration using the Kafka balancing tool, if the execution progress of the balancing tool is blocked due to a Broker fault in the cluster, you need to manually rectify the fault. The scenarios are as follows:

- The Broker is faulty because the disk usage reaches 100%.
  a. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka** > **Instance**, stop the Broker instance in the **Restoring** state, and record the management IP address of the node where the instance resides and the corresponding **broker.id**. You can click the role name to view the value, on the **Instance Configurations** page, select **All Configurations** and search for the **broker.id** parameter.
  b. Log in to the recorded management IP address as user **root**, and run the **df -lh** command to view the mounted directory whose disk usage is 100%, for example, **${BIGDATA_DATA_HOME}/kafka/data1**.
  c. Go to the directory, run the **du -sh *** command to view the size of each file in the directory, Check whether files other than files in the **kafka-logs** directory exist, and determine whether these files can be deleted or migrated.
     - If yes, delete or migrate the related data and go to **8**.
     - If no, go to **4**.
  d. Go to the **kafka-logs** directory, run the **du -sh *** command, select a partition folder to be moved. The naming rule is **Topic name-Partition ID**. Record the topic and partition.
  e. Modify the **recovery-point-offset-checkpoint** and **replication-offset-checkpoint** files in the **kafka-logs** directory in the same way.
     i. Decrease the number in the second line in the file. (To remove multiple directories, the number deducted is equal to the number of files to be removed.)
     ii. Delete the line of the to-be-removed partition. (The line structure is "*Topic name Partition ID Offset*". Save the data before deletion. Subsequently, the content must be added to the file of the same name in the destination directory.)
  f. Modify the **recovery-point-offset-checkpoint** and **replication-offset-checkpoint** files in the destination data directory (for example, **${BIGDATA_DATA_HOME}/kafka/data2/kafka-logs**) in the same way.
     - Increase the number in the second line in the file. (To move multiple directories, the number added is equal to the number of files to be moved.)
     - Add the to-be moved partition to the end of the file. (The line structure is "*Topic name Partition ID Offset*". You can copy the line data saved in **5**.)
  g. Move the partition to the destination directory. After the partition is moved, run the **chown omm:wheel -R** *Partition directory* command to modify the directory owner group for the partition.
  h. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Kafka** > **Instance** to start the stopped Broker instance.
  i. Wait for 5 to 10 minutes and check whether the health status of the Broker instance is **Good**.

    - If yes, resolve the disk capacity insufficiency problem according to the handling method of "ALM-38001 Insufficient Kafka Disk Capacity" after the alarm is cleared.
    - If no, contact O&M support.

  After the faulty Broker is recovered, the blocked balancing task continues. You can run the **--status** command to view the task execution progress.

- The Broker fault occurs because of other causes, the fault scenario is clear, and the fault can be rectified within a short period of time.
  a. Restore the faulty Broker according to the root cause.
  b. After the faulty Broker is recovered, the blocked balancing task continues. You can run the **--status** command to view the task execution progress.
- The Broker fault occurs because of other causes, the fault scenario is complex, and the fault cannot be rectified within a short period of time.
  a. Run the **kinit** *Kafka administrator account* command (skip this step in normal mode).
  b. Run the **zkCli.sh -server <***ZooKeeper cluster service IP address:zkPort***/kafka>** command to log in to ZooKeeper Shell.
  c. Run the **addauth krbgroup** command (skip this step in normal mode).
  d. Delete the **/admin/reassign_partitions** and **/controller** directories.
  e. Perform the preceding steps to forcibly stop the migration. After the cluster recovers, run the **kafka-reassign-partitions.sh** command to delete redundant copies generated during the intermediate process.

18.7.4 Configuring the Path for Extranet Clients to Access Kafka Broker

This section applies to MRS 3.2.0 or later.

Scenario

To access Kafka Broker deployed on a private network from the Kafka client via the Internet, enable the Kafka private and public network traffic distribution function.

Prerequisites

- The node where Broker resides has both private and public IP addresses. Broker is bound to the private IP address and cannot be accessed via the Internet. Alternatively, the node where Broker resides has only private IP addresses, and external services access the private network through gateway mapping.
- The ZooKeeper service is running properly.
- The Kafka instance status and disk status are normal.

Procedure

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **Kafka**. On the page that is displayed, click the **Instance** tab. On this tab page, select the target broker instance in the instance list. On the displayed page, click the **Instance Configurations** tab and then the **All Configurations** sub-tab. Enter **broker.id** in the search box to view and record the broker ID of the current broker instance.

Step 3 Repeat **Step 2** to view and record the broker ID of each broker instance.

Step 4 Choose **Cluster** > **Services** > **Kafka**. On the page that is displayed, click the **Configurations** tab then the **All Configurations** sub-tab. On this sub-tab page, click **Broker(Role)** and select **Server**. Enter **advertised** and **actual** in the search box. The five configuration items shown in the following figure are displayed. Configure the parameters according to **Table 18-8**.

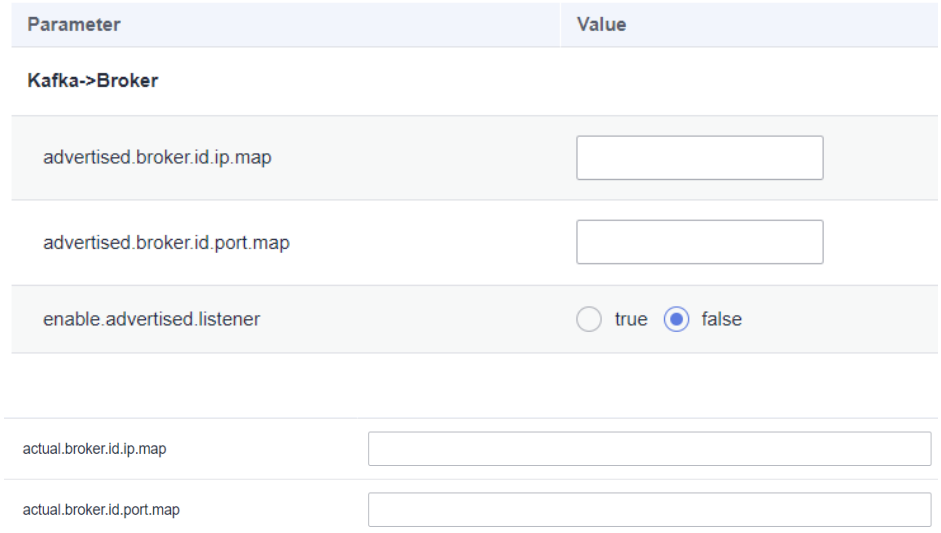


Table 18-8 Parameter description

Parameter	Description	Remarks
enable.advertised.listener	Whether to enable the advertised.listeners configuration. The default value is **false**.	Set **enable.advertised.listener** to **true**. **NOTE** When you install the Kafka service, do not set this parameter to **true**. You can set this parameter to **true** only after broker instances and ZooKeeper are running properly.

Parameter	Description	Remarks
advertised.broker.id.ip.map	IP address released by Kafka. This parameter is left blank by default. Format: *Broker ID:IP*. Multiple brokers can use the same IP address. If multiple mappings are configured, use commas (,) to separate them.	Map the broker ID of each broker instance recorded in **Step 2** to the IP address to which the broker instance to bound. For example, if there are three broker instances and one IP address, the broker IDs are **1**, **2**, and **3**, and the IP address is **10.***xxx.xxx.xxx*, the configuration format is **1:10.***xxx.xxx.xxx***,2:10.***xxx.xxx.xxx***,3:10.***xxx.xxx.xxx*.
advertised.broker.id.port.map	Port released by Kafka. This parameter is left blank by default. Format: *Broker ID:Port*. **Port** indicates the port to be bound. This port is a custom port and must be available. If multiple mappings are configured, use commas (,) to separate them.	Map the broker ID of each broker instance recorded in **Step 2** to the port to which the broker instance to bound. For example, if there are three broker instances and three ports, the broker IDs are **1**, **2**, and **3**, and the ports are **3307**, **3308**, and **3309**, the configuration format is **1:3307,2:3308,3:3309**.
actual.broker.id.ip.map	IP address bound to Kafka. This parameter is left blank by default. Format: *Broker ID:IP*. If multiple mappings are configured, use commas (,) to separate them.	Map the broker ID of each broker instance recorded in **Step 2** to the IP address to which the broker instance to bound. For example, if there are three broker instances and one IP address, the broker IDs are **1**, **2**, and **3**, and the IP address is **10.***xxx.xxx.xxx*, the configuration format is **1:10.***xxx.xxx.xxx***,2:10.***xxx.xxx.xxx***,3:10.***xxx.xxx.xxx*.
actual.broker.id.port.map	Port bound to Kafka. This parameter is left blank by default. Format: *Broker ID:Port*. **Port** indicates the port to be bound. This port is a custom port and must be available. If multiple mappings are configured, use commas (,) to separate them.	Map the broker ID of each broker instance recorded in **Step 2** to the port to which the broker instance to bound. For example, if there are three broker instances and three ports, the broker IDs are **1**, **2**, and **3**, and the ports are **3307**, **3308**, and **3309**, the configuration format is **1:3307,2:3308,3:3309**.

Step 5 After the configuration is complete, click **Save** in the upper left corner. On the **Instance** tab, select the target broker instances and choose **More** > **Instance Rolling Restart**. Wait until the rolling restart is complete.

Step 6 (Optional) To disable this configuration, set **enable.advertised.listener** to **false** and click **Save**. On the **Instance** page of Kafka, select Kafka instances, choose **More** > **Instance Rolling Restart**, and wait until the rolling restart is complete.

----End

NOTE

- In a cluster with Kerberos authentication enabled, after **enable.advertised.listener** is configured, the client supports only Kerberos authentication, but not PLAIN authentication.
- The **Port** parameter in **advertised.broker.id.port.map** and **actual.broker.id.port.map** can be set to the same value.

18.8 Kafka Performance Tuning

Scenario

You can modify Kafka server parameters to improve Kafka processing capabilities in specific service scenarios.

Parameter Tuning

Modify the service configuration parameters. For details, see **Modifying Cluster Service Configuration Parameters**. For details about the tuning parameters, see **Table 18-9**.

Table 18-9 Tuning parameters

Parameter	Default Value	Scenario
num.recovery.threads.per.data.dir	10	During the Kafka startup process, if a large volume of data exists, you can increase the value of this parameter to accelerate the startup.
background.threads	10	Specifies the number of threads processed by a broker background task. If a large volume of data exists, you can increase the value of this parameter to improve broker processing capabilities.

Parameter	Default Value	Scenario
num.replica.fetchers	1	Specifies the number of threads used when a replica requests to the Leader for data synchronization. If the value of this parameter is increased, the replica I/O concurrency increases.
num.io.threads	8	Specifies the number of threads used by the broker to process disk I/O. It is recommended that the number of threads be greater than or equal to the number of disks.
KAFKA_HEAP_OPTS	-Xmx6G -Xms6G	Specifies the Kafka JVM heap memory setting. If the data volume on the broker is large, adjust the heap memory size.

18.9 Kafka O&M Management

18.9.1 Kafka Common Configuration Parameters

This section applies to MRS 3.x or later.

Navigation path for setting parameters:

For details about how to set parameters, see **Modifying Cluster Service Configuration Parameters**.

Common Parameters

Table 18-10 Parameter description

Parameter	Description	Default Value
log.dirs	List of Kafka data storage directories. Use commas (,) to separate multiple directories.	%{@auto.detect.datapart.bk.log.logs}
KAFKA_HEAP_OPTS	Specifies the JVM option used for Kafka to start broker. It is recommended that you set this parameter based on service requirements.	-Xmx6G -Xms6G

Parameter	Description	Default Value
auto.create.topics.enable	Indicates whether a topic is automatically created. If this parameter is set to **false**, you need to run a command to create a topic before sending a message.	true
default.replication.factor	Default number of replicas of a topic is automatically created.	2
monitor.preInitDelay	Delay of the first health check after the server is started. If the startup takes a long time, increase the value of the parameter. Unit: millisecond	600,000

Timeout Parameters

Table 18-11 Broker-related timeout parameters

Parameter	Description	Default Value	Impact
controller.socket.timeout.ms	Specifies the timeout for connecting controller to broker. Unit: millisecond	30,000	Generally, retain the default value of this parameter.
group.max.session.timeout.ms	Specifies the maximum session timeout during the consumer registration. Unit: millisecond	1800000	The configured value must be less than the value of this parameter.
group.min.session.timeout.ms	Specifies the minimum session timeout during the consumer registration. Unit: millisecond	6,000	The configured value must be greater than the value of this parameter.
offsets.commit.timeout.ms	Specifies the timeout for the Offset to submit requests. Unit: millisecond	5,000	This parameter specifies the maximum delay for processing an Offset request.

Parameter	Description	Default Value	Impact
replica.socket.timeout.ms	Specifies the timeout of the request for synchronizing replica data. Its value must be greater than or equal to that of the **replica.fetch.wait.max.ms** parameter. Unit: millisecond	30,000	Specifies the maximum timeout for establishing a channel before the synchronization thread sends a synchronization request. The value must be greater than that of the **replica.fetch.wait.max.ms** parameter.
request.timeout.ms	Specifies the timeout for waiting for a response after the client sends a connection request. Unit: millisecond	30,000	This parameter is configured when the networkclient connection is transferred in the controller and replica threads on the broker node.If no response is received within the timeout, the client resends the request. A request failure is returned after the maximum retry times is reached.
transaction.max.timeout.ms	Specifies the maximum timeout allowed by the transaction. Unit: millisecond	900,000	Specifies the maximum timeout for transactions.If the client request time exceeds the value of this parameter, broker returns an error in InitProducerIdRequest. This prevents a long client request timeout, ensuring that consumer can receive topics.

Parameter	Description	Default Value	Impact
user.group.cache.timeout.sec	Specifies the time when the user group information is stored in the cache. Unit: second	300	Specifies the time for caching the mapping between users and user groups. If time exceeds the threshold, the system automatically runs the **id -Gn** command to query the user information. During this period, the mapping in the cache is used.
zookeeper.connection.timeout.ms	Specifies the timeout for connecting to ZooKeeper. Unit: millisecond	45,000	This parameter specifies the duration for connecting the ZooKeeper and zkclient for the first time. If the duration exceeds the value of this parameter, the zkclient automatically disconnects the connection.

Parameter	Description	Default Value	Impact
zookeeper.session.timeout.ms	Specifies the ZooKeeper session timeout duration. During this period, ZooKeeper disconnects the connection if broker does not report its heartbeats to ZooKeeper. Unit: millisecond	45,000	ZooKeeper session timeout has the following functions: 1) Based on value of this parameter and the number of ZooKeeper URLs in ZKURL, if the connection duration exceeds the node timeout value (sessionTimeout/ Number of transferred ZooKeeper URLs), the connection fails and the system attempts to connect to the next node. 2) After the connection is established, a session (for example, the temporary BrokerId node registered on the ZooKeeper) is cleared by the ZooKeeper a session timeout later if the broker is stopped.

Table 18-12 Producer-related timeout parameters

Parameter	Description	Default Value	Impact
request.timeout.ms	Specifies the timeout of a message request.Unit: millisecond	30,000	If a network fault occurs, increase the value of this parameter. If the value is too small, the Batch Expire occurs.

Table 18-13 Consumer-related timeout parameters

Parameter	Description	Default Value	Impact
connections.max.idle.ms	Specifies the maximum retention period for idle connections. Unit: millisecond	600,000	If the idle connection time is greater than this parameter value, this connection is disconnected. If necessary, a new connection is created.
request.timeout.ms	Specifies the timeout for consumer requests.Unit: millisecond	30,000	If the request times out, the request will fail and be sent again.

18.9.2 Kafka Log Overview

Log Description

Log paths: The default storage path of Kafka logs is **/var/log/Bigdata/kafka**. The default storage path of audit logs is **/var/log/Bigdata/audit/kafka**.

- Broker: **/var/log/Bigdata/kafka/broker** (run logs)
- Kafka UI: **/var/log/Bigdata/kafka/ui** (run logs)
- MirrorMaker: **/var/log/Bigdata/kafka/mirrormaker** (run logs)

Log archive rule: The automatic Kafka log compression function is enabled. By default, when the size of logs exceeds 30 MB, logs are automatically compressed into a log file named in the following format: *<Original log file name>-<yyyy-mm-dd_hh-mm-ss>.[ID]*.**log.zip**. A maximum of 20 latest compressed files are retained by default. You can configure the number of compressed files and the compression threshold.

Table 18-14 Broker log list

Type	Log File Name	Description
Run log	server.log	Server run log of the broker process
	controller.log	Controller run log of the broker process
	kafka-request.log	Request run log of the broker process
	log-cleaner.log	Cleaner run log of the broker process

Type	Log File Name	Description
	state-change.log	State-change run log of the broker process
	kafkaServer-<SSH_USER>-<DATE>-<PID>-gc.log	GC log of the broker process
	postinstall.log	Work log after broker installation
	prestart.log	Work log before broker startup
	checkService.log	Log that records whether broker starts successfully
	start.log	Startup log of the broker process
	stop.log	Stop log of the broker process
	checkavailable.log	Log that records the health check details of the Kafka service
	checkInstanceHealth.log	Log that records the health check details of broker instances
	kafka-authorizer.log	Broker authorization log
	kafka-root.log	Broker basic log
	cleanup.log	Cleanup log of broker uninstallation
	metadata-backup-recovery.log	Broker backup and recovery log
	ranger-kafka-plugin-enable.log	Log that records the Ranger plug-ins enabled by brokers
	server.out	Broker JVM log
	audit.log	Authentication log of the Ranger authentication plug-in. This log is archived in the **/var/log/Bigdata/audit/kafka** directory.

Table 18-15 Kafka UI log list

Type	Log File Name	Description
Run log	kafka-ui.log	Run log of the Kafka UI process
	postinstall.log	Work log after Kafka UI installation
	cleanup.log	Cleanup log of Kafka UI uninstallation
	prestart.log	Work log before Kafka UI startup
	ranger-kafka-plugin-enable.log	Log that records the Ranger plug-ins enabled by Kafka UI
	start.log	Startup log of the Kafka UI process
	stop.log	Stop log of the Kafka UI process
	start.out	Kafka UI process startup information
Audit log	audit.log	Audit log of the KafkaUI service
Authentication log	kafka-authorizer.log	Run log file of the open-source authentication plug-in of Kafka. This log is archived in the **/var/log/Bigdata/audit/kafka/kafkaui** directory.
	ranger-authorizer.log	Run log of the Ranger authentication plug-in. This log is archived in the **/var/log/Bigdata/audit/kafka/kafkaui** directory.

Table 18-16 MirrorMaker log list

Type	Log File Name	Description
Run log	mirrormaker.out	MirrorMaker process startup information

Type	Log File Name	Description
	mirrormaker.log	Run log of the MirrorMaker process
	cleanup.log	Cleanup log of MirrorMaker uninstallation
	prestart.log	Work log before MirrorMaker startup
	start.log	Startup log of the MirrorMaker process
	postinstall.log	Work log after MirrorMaker installation
	stop.log	Stop log of the MirrorMaker process
	mirrorMaker-omm-***-pid***-gc.log.*.current	MirrorMaker process GC log

Log Level

Table 18-17 describes the log levels supported by Kafka.

Levels of run logs are ERROR, WARN, INFO, and DEBUG from the highest to the lowest priority. Run logs of equal or higher levels are recorded. The higher the specified log level, the fewer the logs recorded.

Table 18-17 Log levels

Level	Description
ERROR	Logs of this level record error information about system running.
WARN	Logs of this level record exception information about the current event processing.
INFO	Logs of this level record normal running status information about the system and events.
DEBUG	Logs of this level record the system information and system debugging information.

To modify log levels, perform the following operations:

Step 1 Go to the **All Configurations** page. See **Modifying Cluster Service Configuration Parameters**.

Step 2 On the menu bar on the left, select the log menu of the target role.

Step 3 Select a desired log level.

Step 4 Save the configuration. In the displayed dialog box, click **OK** to make the configurations take effect.

----End

Log Format

The following table describes the Kafka log format.

Table 18-18 Log formats

Type	Format	Example
Run log	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<Thread that generates the log>|<Message in the log>|<Full name of the log event invocation class>(<Log file>:<Row>)	2015-08-08 11:09:53,483 | INFO | [main] | Loading logs. | kafka.log.LogManager (Logging.scala:68)
	<yyyy-MM-dd HH:mm:ss><HostName><Component name><logLevel><Message>	2015-08-08 11:09:51 10-165-0-83 Kafka INFO Running kafka-start.sh.

18.9.3 Changing the Broker Storage Directory

Scenario

This section applies to MRS 3.x or later.

When a broker storage directory is added, the MRS cluster administrator needs to change the broker storage directory on FusionInsight Manager, to ensure that the Kafka can work properly. The new topic partition will be generated in the directory that has fewest partitions. Changing the ZooKeeper storage directory includes the following scenarios:

NOTE

Because Kafka does not detect disk capacity, ensure that the disk quantity and capacity configured for each Broker instance are the same.

- Change the storage directory of the Broker role. In this way, the storage directories of all Broker instances are changed.
- Change the storage directory of a single Broker instance. In this way, only the storage directory of this Broker instance is changed, and the storage directories of other Broker instances remain the same.

Impact on the System

- Changing the Broker role storage directory requires the restart of services. The services cannot be accessed during the restart.
- The storage directory of a single Broker instance can be changed only after the instance is restarted. The instance cannot provide services during the restart.
- The directory for storing service parameter configurations must also be updated.

Prerequisites

- New disks have been prepared and installed on each data node, and the disks are formatted.
- The Kafka client has been installed.
- When you change the storage directory of a single Broker instance, the number of active Broker instances must be greater than the number of backups specified during topic creation.

Procedure

Changing the storage directory of the Kafka role

Step 1 Log in as user **root** to each node on which the Kafka service is installed, and perform the following operations:

1. Create a target directory.

   For example, to create the target directory **${BIGDATA_DATA_HOME}/kafka/data2**, run the following command:

   mkdir ${BIGDATA_DATA_HOME}/kafka/data2

2. Mount the directory to the new disk. For example, mount **${BIGDATA_DATA_HOME}/kafka/data2** to the new disk.
3. Modify permissions on the new directory.

   For example, to modify permissions on the **${BIGDATA_DATA_HOME}/kafka/data2** directory, run the following commands:

   chmod 700 ${BIGDATA_DATA_HOME}/kafka/data2 -R and **chown omm:wheel ${BIGDATA_DATA_HOME}/kafka/data2 -R**

Step 2 Log in to FusionInsight Manager for clusters of MRS 3.*x* or later and choose **Cluster** > **Services** > **Kafka** > **Configurations**.

Step 3 Add a new directory to the end of the default value of **log.dirs**.

Enter **log.dirs** in the search box and add the new directory to the end of the default value of the **log.dirs** configuration item. Use commas (,) to separate multiple directories. For example:

${BIGDATA_DATA_HOME}/kafka/data1/kafka-logs,${BIGDATA_DATA_HOME}/kafka/data2/kafka-logs

Step 4 Click **Save**, and then click **OK**. When **Operation succeeded** is displayed, click **Finish**.

Step 5 Choose **Cluster** > **Services** > **Kafka**. In the upper right corner, choose **More** > **Restart Service** to restart the Kafka service.

Changing the storage directory of a single Kafka instance

Step 6 Log in to the Broker node as user **root** and perform the following operations:

1. Create a target directory.

   For example, to create the target directory **${BIGDATA_DATA_HOME}/kafka/data2**, run the following command:

   mkdir ${BIGDATA_DATA_HOME}/kafka/data2

2. Mount the directory to the new disk. For example, mount **${BIGDATA_DATA_HOME}/kafka/data2** to the new disk.
3. Modify permissions on the new directory.

   For example, to modify permissions on the **${BIGDATA_DATA_HOME}/kafka/data2** directory, run the following commands:

   chmod 700 ${BIGDATA_DATA_HOME}/kafka/data2 -R and **chown omm:wheel ${BIGDATA_DATA_HOME}/kafka/data2 -R**

Step 7 Log in to FusionInsight Manager for MRS 3.*x* or later, and choose **Cluster** > **Services** > **Kafka** > **Instance**.

Step 8 Click the specified broker instance and switch to **Instance Configurations**.

Enter **log.dirs** in the search box and add the new directory to the end of the default value of the **log.dirs** configuration item. Use commas (,) to separate multiple directories, for example, **${BIGDATA_DATA_HOME}/kafka/data1/kafka-logs,${BIGDATA_DATA_HOME}/kafka/data2/kafka-logs**.

Step 9 Click **Save**, and then click **OK**. A message is displayed, indicating that the operation is successful. Click **Finish**.

Step 10 On the Broker instance page, choose **More** > **Restart Instance** to restart the Broker instance.

----End

18.9.4 Migrating Data Between Kafka Nodes

Scenario

You can run Kafka client commands to migrate data between partitions on a node without stopping services. You can also use the Kafka UI to migrate partitions.

Prerequisites

- The MRS cluster administrator has understood service requirements and prepared a Kafka user (belonging to the **kafkaadmin** group. It is not required for the normal mode.).
- The Kafka client has been installed.
- The Kafka instance status and disk status are normal.
- Based on the current disk space usage of the partition to be migrated, ensure that the disk space will be sufficient after the migration.

Migrating Data Using the Kafka Client

Step 1 Log in as a client installation user to the node on which the Kafka client is installed.

Step 2 Run the following command to switch to the Kafka client installation directory, for example, **/opt/kafkaclient**:

cd /opt/kafkaclient

Step 3 Run the following command to set environment variables:

source bigdata_env

Step 4 Run the following command to authenticate the user (skip this step in normal mode):

kinit *Component service user*

Step 5 Run the following command to switch to the Kafka client directory:

cd Kafka/kafka/bin

Step 6 Run the following command to view the topic details of the partition to be migrated:

Security mode:

./kafka-topics.sh --describe --bootstrap-server *IP address of the Kafkacluster:21007* **--command-config ../config/client.properties --topic** *topic name*

Normal mode:

./kafka-topics.sh --describe --bootstrap-server *IP address of the Kafka cluster:21005* **--command-config ../config/client.properties --topic** *Topic name*

Step 7 Run the following command to query the mapping between **Broker_ID** and the IP address:

./kafka-broker-info.sh --zookeeper *IP address of the ZooKeeper quorumpeer instance*:*ZooKeeper port number*/**kafka**

```
Broker_ID     IP_Address
--------------------------
4             192.168.0.100
```



```
5       192.168.0.101
6       192.168.0.102
```

NOTE

- IP address of the ZooKeeper quorumpeer instance

  To obtain IP addresses of all ZooKeeper quorumpeer instances, log in to FusionInsight Manager and choose **Cluster** > **Services** > **ZooKeeper**. On the displayed page, click **Instance** and view the IP addresses of all the hosts where the quorumpeer instances locate.

- Port number of the ZooKeeper client

  Log in to FusionInsight Manager and choose **Cluster** > **Service** > **ZooKeeper**. On the displayed page, click **Configurations** and check the value of **clientPort**. The default value is **24002**.

Step 8 Obtain the partition distribution and node information from the command output in **Step 6** and **Step 7**, and create the JSON file for reallocation in the current directory.

To migrate data in the partition whose **Broker_ID** is **6** to the **/srv/BigData/hadoop/data1/kafka-logs** directory, the required JSON configuration file is as follows:

```
{"partitions":[{"topic": "testws","partition": 2,"replicas": [6,5],"log_dirs": ["/srv/BigData/hadoop/data1/kafka-logs","any"]}],"version":1}
```

NOTE

- **topic** indicates the topic name, for example, **testws**.
- **partition** indicates the topic partition.
- The number in **replicas** corresponds to **Broker_ID**.
- **log_dirs** indicates the path of the disk to be migrated. In this example, **log_dirs** of the node whose **Broker_ID** is **5** is set to **any**, and that of the node whose **Broker_ID** is **6** is set to **/srv/BigData/hadoop/data1/kafka-logs.** Note that the path must correspond to the node.

Step 9 Run the following command to perform reallocation:

Security mode:

./kafka-reassign-partitions.sh --bootstrap-server *Service IP address of Broker*:**21007 --command-config ../config/client.properties --reassignment-json-file** *Path of the JSON file compiled in **Step 8*** **--execute**

Normal mode:

./kafka-reassign-partitions.sh --bootstrap-server *Service IP address of Broker*:**21005 --command-config ../config/client.properties --reassignment-json-file** *Path of the JSON file compiled in **Step 8*** **--execute**

If message "Successfully started reassignment of partitions" is displayed, the execution is successful.

----End

Migrating Partitions Using KafkaUI (Versions earlier than MRS 3.5.0)

Step 1 Access the Kafka UI.

1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

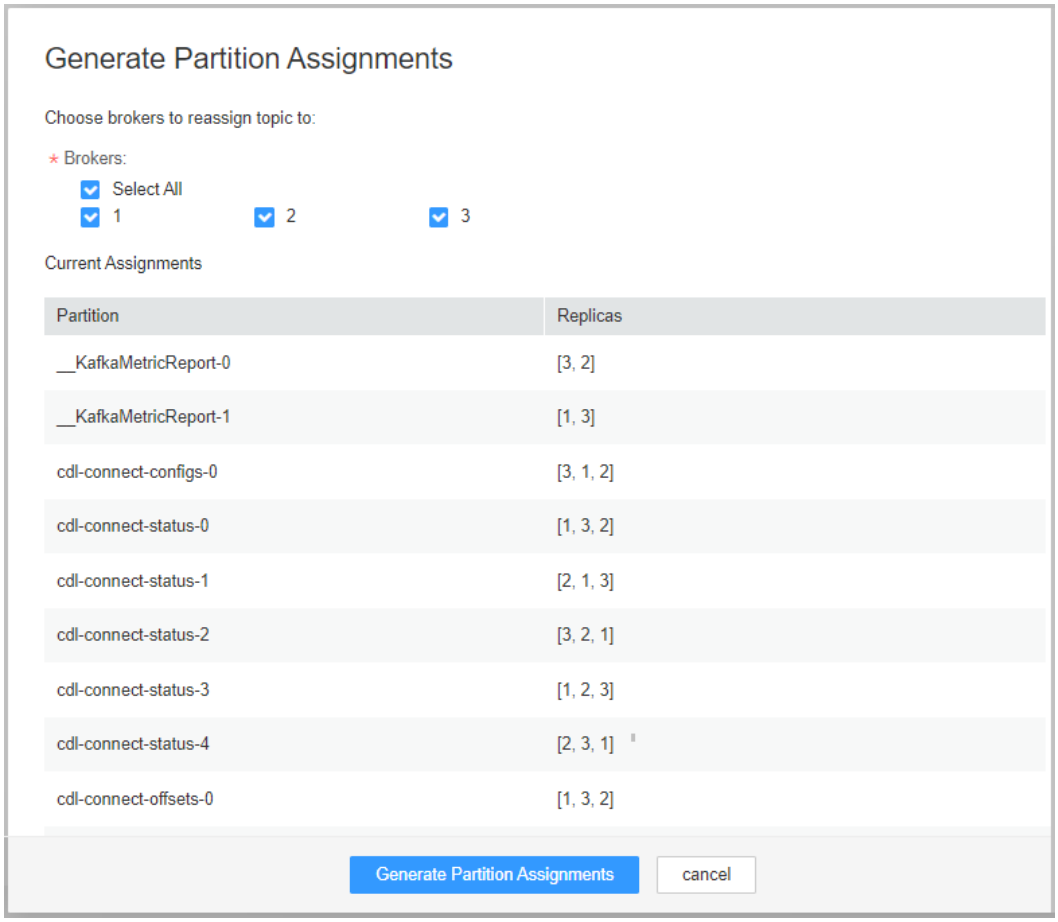
If you need to perform related operations on the page, for example, creating a topic, you need to grant related permissions to the user. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access Kafka UI.

Step 2 Click **Generate assignment**. The **Generate Partition Assignments** page is displayed.

Step 3 In the **Brokers** area, select brokers to which the topic is to be re-assigned.

Step 4 Click **Generate Partition Assignments** to generate a partition migration solution.



Step 5 Click **Run assignment** to migrate a partition.

----End

Migrating Partitions Using KafkaUI (Versions later than MRS 3.5.0)

Step 1 Access the Kafka UI.

1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

   If you need to perform related operations on the page, for example, creating a topic, you need to grant related permissions to the user. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access Kafka UI.

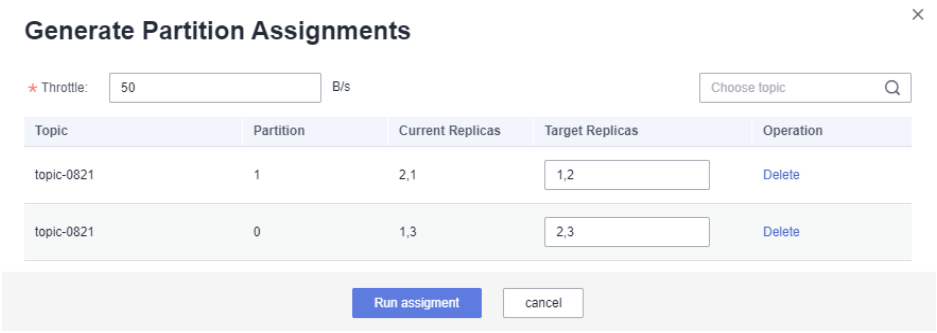
Step 2 Click **Generate assignment**. The **Generate Partition Assignments** page is displayed.

Step 3 In the **Brokers** area, select brokers to which the topic is to be re-assigned.

Step 4 In the search box, search for the topic and partition to be migrated.

- The **Partition** column displays the topics to be migrated. The naming rule is *Topic name-Partitioned index*.
- The **Replicas** column displays the broker IDs where the topic partition is located.

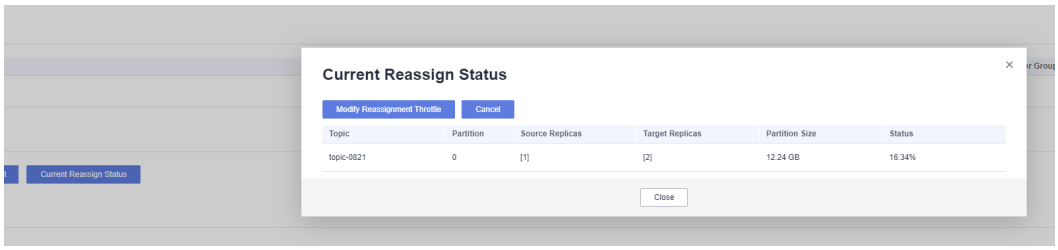
Step 5 Click **Generate Partition Assignments** to generate a partition migration plan.



- **Throttle**: You can configure the flow control parameter for partition migration. If you leave it unconfigured, the rate will not be limited, as the default value is **0**.
- **Target Replicas**: automatically generated partition migration plan, which can be manually modified. The format is *Broker ID 1*,*Broker ID 2*.
- **Operation**: You can delete topics that do not need to be migrated.

Step 6 Click **Run assignment** to execute the partition migration plan.

Step 7 Click **Current Reassign Status** to view the execution status of the current partition migration plan.



- Click **Modify Reassignment Throttle** to modify flow control parameters for zone migration.
- Click **Cancel** to cancel all partition migration plans that are being executed.

----End

18.9.5 Using the Kafka Balancing Tool to Limit the Production and Consumption Speed

Scenario

Use the **kafka-configs.sh** command-line tool on the client side to adjust Kafka cluster settings in line with service demands. This tool also enables the regulation of Kafka message throughput at various scopes: topic, user, and client levels.

NOTE

This function is available in MRS 3.3.1 and later versions.

Prerequisites

- The MRS cluster administrator has learned service requirements. You have created a Kafka component service user that belongs to the **kafkaadmin** user group (this is not required for normal clusters).
- The Kafka client has been installed in a directory, for example, **/opt/client**.

Procedure

Step 1 Log in as a client installation user to the node on which the Kafka client is installed.

Step 2 Switch to the Kafka client installation directory, for example, **/opt/client**.

cd /opt/client

Step 3 Configure environment variables.

source bigdata_env

Step 4 Authenticate the user (skip this step for normal clusters).

kinit *Component service user*

Step 5 Switch to the Kafka client installation directory.

cd Kafka/kafka

Step 6 Use **kafka-configs.sh** to manage Kafka traffic. Below are the commonly used commands.

NOTE

The subsequent section details the retrieval process for certain command parameters. The actual parameter values should be used when available.

- **Service IP address of any ZooKeeper node**: Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ZooKeeper** and click **Instance**. You can choose the service IP address of any ZooKeeper instance, for example, **192.168.20.36**.
- **clientPort**: Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ZooKeeper**, click **Configurations**, and click **All Configurations**. On the displayed page, search for and record the value of **clientPort**, for example, **24002**.
- **Kafka cluster IP address**: Log in to FusionInsight Manager, click **Services**, click **Kafka**, and click **Instance**. You can choose the service IP address of any Broker instance, for example, **192.168.20.36**.
- The IP port ID is **21007** for Kafka clusters with Kerberos authentication enabled. For normal Kafka clusters, the IP port ID is **9092**.
- **Client ID**: Log in to the Kafka client, run the following command to obtain the value of **CLIENT-ID**. For example, the obtained client ID is **clientA**.

  bin/kafka-consumer-groups.sh --describe --bootstrap-server *Kafka cluster IP address:Port* **--all-groups --command-config config/consumer.properties**

- Topic-level production traffic limiting

  bin/kafka-configs.sh --zookeeper *Service IP address of any ZooKeeper node:clientPort*/**kafka --alter --add-config** '**producer_byte_rate=***Production traffic limiting speed*' **--entity-type topics_limit --entity-name** *Topic name*

  The following is an example:

  bin/kafka-configs.sh --zookeeper 192.168.20.36:24002/kafka --alter --add-config 'producer_byte_rate=10485760' --entity-type topics_limit --entity-name testTopic-01

- Topic-level consumption rate limiting

  bin/kafka-configs.sh --zookeeper *Service IP address of any ZooKeeper node:clientPort*/**kafka --alter --add-config** '**consumer_byte_rate=***Consumption traffic limiting speed*' **--entity-type topics_limit --entity-name** *Topic name*

  The following is an example:

  bin/kafka-configs.sh --zookeeper 192.168.20.36:24002/kafka --alter --add-config 'consumer_byte_rate=1048576' --entity-type topics_limit --entity-name testTopic-01

- User-level production traffic limiting

  bin/kafka-configs.sh --bootstrap-server *Kafka cluster IP address:Port number* **--alter --add-config** '**producer_byte_rate=***Production traffic limiting speed*' **--entity-type users --entity-name** *Component service user* **--command-config config/client.properties**

  The following is an example:

  bin/kafka-configs.sh --bootstrap-server 192.168.20.36:21007 --alter --add-config 'producer_byte_rate=10485760' --entity-type users --entity-name kafkauser --command-config config/client.properties

- User-level consumption traffic limiting

  bin/kafka-configs.sh --bootstrap-server *Kafka cluster IP address:Port number* **--alter --add-config** '**consumer_byte_rate=***Consumption traffic*

limiting speed' **--entity-type users --entity-name** *Component service user* **--command-config config/client.properties**

The following is an example:

bin/kafka-configs.sh --bootstrap-server 192.168.20.36:21007 --alter --add-config 'consumer_byte_rate=1048576' --entity-type users --entity-name kafkauser --command-config config/client.properties

- Client-level production traffic limiting

  bin/kafka-configs.sh --bootstrap-server *Kafka cluster IP address:Port number* **--alter --add-config** '**producer_byte_rate=***Production traffic limiting speed*' **--entity-type clients --entity-name** *Client ID* **--command-config config/client.properties**

  The following is an example:

  bin/kafka-configs.sh --bootstrap-server 192.168.20.36:21007 --alter --add-config 'producer_byte_rate=10485760' --entity-type clients --entity-name clientA --command-config config/client.properties

- Client-level consumption traffic limiting

  bin/kafka-configs.sh --bootstrap-server *Kafka cluster IP address:Port number* **--alter --add-config 'consumer_byte_rate=***Consumption traffic limiting speed*' **--entity-type clients --entity-name** *Client ID* **--command-config config/client.properties**

  The following is an example:

  bin/kafka-configs.sh --bootstrap-server 192.168.20.36:21007 --alter --add-config 'consumer_byte_rate=1048576' --entity-type clients --entity-name clientA --command-config config/client.properties

- Combined user and client production and consumption traffic limiting

  bin/kafka-configs.sh --bootstrap-server *Kafka cluster IP address:Port number* **--alter --add-config 'producer_byte_rate=***Production traffic limiting speed***,consumer_byte_rate=***Consumption traffic limiting speed*' **--entity-type users --entity-name** *Component service user* **--entity-type clients --entity-name** *Client ID* **--command-config config/client.properties**

  The following is an example:

  bin/kafka-configs.sh --bootstrap-server 192.168.20.36:21007 --alter --add-config 'producer_byte_rate=1048576,consumer_byte_rate=1048576' --entity-type users --entity-name kafkauser --entity-type clients --entity-name clientD --command-config config/client.properties

- Viewing traffic limiting information

  bin/kafka-configs.sh --describe --bootstrap-server *Kafka cluster IP address:Port number* **--entity-type** *myType* **--entity-name** *myName* **--command-config config/client.properties**

  NOTE

  - The permissible values for **myType** are **topics_limit**, **users**, and **clients**, and those for **myName** are **Topic name**, **Component service user**, and **Client ID**.
  - **producer_byte_rate** and **consumer_byte_rate** indicate the production rate and consumption rate, respectively. The unit is byte/s.
  - If **myType** is set to **topics_limit**, change the command to the following:

    bin/kafka-configs.sh --describe --zookeeper *Service IP address of any ZooKeeper node:clientPort***/kafka --entity-type topics_limit --entity-name** *Topic name*

The following is an example:

bin/kafka-configs.sh --describe --bootstrap-server 192.168.20.36:21007 --entity-type users --entity-name kafkauser --entity-type clients --entity-name clientD --command-config config/client.properties

- Canceling traffic limiting

  bin/kafka-configs.sh --bootstrap-server *Kafka cluster IP address:Port number* **--alter --delete-config 'producer_byte_rate,consumer_byte_rate' --entity-type** *myType* **--entity-name** *myName* **--command-config config/client.properties**

  NOTE

  - The permissible values for **myType** are **topics_limit**, **users**, and **clients**, and those for **myName** are **Topic name**, **Component service user**, and **Client ID**.
  - If **myType** is set to **topics_limit**, change the command to the following:

    bin/kafka-configs.sh --zookeeper *Service IP address of any ZooKeeper node:clientPort*/**kafka --alter --delete-config 'producer_byte_rate,consumer_byte_rate' --entity-type topics_limit --entity-name** *Topic name*

  The following is an example:

  bin/kafka-configs.sh --bootstrap-server 192.168.20.36:21007 --alter --delete-config 'producer_byte_rate,consumer_byte_rate' --entity-type users --entity-name kafkauser --entity-type clients --entity-name client --command-config config/client.properties

----End

18.9.6 Configure Lag Alarm Rules

Scenario

The Kafka UI allows you to view and modify lag alarm rules. If these rules are triggered, alarm "ALM-38018 Kafka Consumer Lag" will be reported.

NOTE

This section applies only to MRS 3.5.0 and later versions.

Configuring Lag Alarm Rules

Step 1 Access the Kafka UI.

1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

   If you need to perform related operations on the page, for example, creating a topic, you need to grant related permissions to the user. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access Kafka UI.

Step 2 Click **Alarms** on the menu bar.

Step 3 Click **Create Lag Alarm**, create a Kafka message lag alarm rule as prompted, and click **Create**.

Figure 18-1 Create Lag Alarm

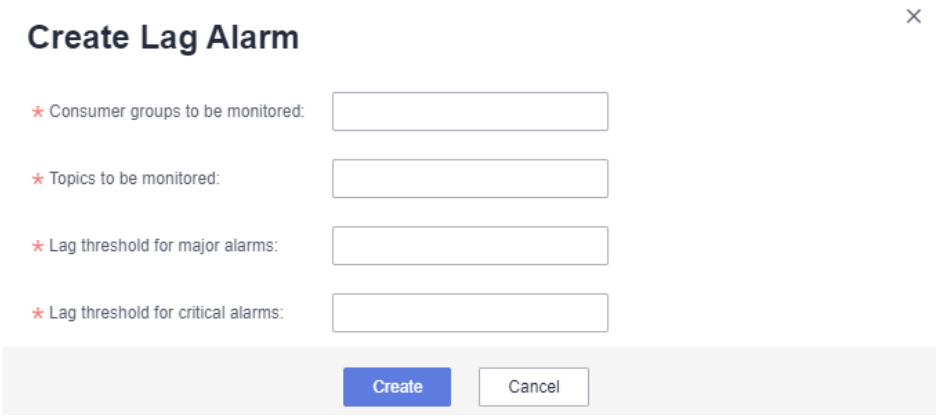
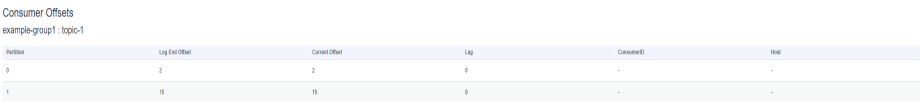


Table 18-19 Lag alarm rule configuration parameters.

Parameter	Description	Example Value
Consumer groups to be monitored	Name of the consumer group to be monitored.	example-group1
Topics to be monitored	Name of the topic to be monitored.	topic-1
Lag threshold for major alarms	Message backlog threshold when a major alarm is sent.	2
Lag threshold for critical alarms	Message backlog threshold when a critical alarm is sent.	4

NOTE

Message lag is calculated as the total lag across all partitions in the current topic. Refer to **Using KafkaUI to View the Current Consumption Status** to check the message lag for all partitions in the topic for the consumer group.



Step 4 You can manage the lag alarm rules by accessing the **Lag Alarms** page, where you can view, modify, or delete them.

----End

18.10 Common Issues About Kafka

18.10.1 Kafka Specifications

This section applies to MRS 3.x or later.

Upper Limit of Topics

The maximum number of topics depends on the number of file handles (mainly used by data and index files on site) opened in the process.

1. Run the **ulimit -n** command to view the maximum number of file handles that can be opened in the process.
2. Run the **lsof -p** *<Kafka PID>* command to view the file handles (which may keep increasing) that are opened in the Kafka process on the current single node.
3. Determine whether the maximum number of file handles will be reached and whether the running of Kafka is affected after required topics are created, and estimate the maximum size of data that each partition folder can store and the number of data (*.log file, whose default size is 1 GB and can be adjusted by modifying **log.segment.bytes**) and index (*.index file, whose default size is 10 MB and can be adjusted by modifying **log.index.size.max.bytes**) files that will be produced after required topics are created.

Number of Concurrent Consumers

In an application, it is recommended that the number of concurrent consumers in a group be the same as the number of partitions in a topic, ensuring that a consumer consumes data in only a specified partition. If the number of concurrent consumers is more than the number of partitions, the redundant consumers have no data to consume.

Relationship Between Topic and Partition

- If *K* Kafka nodes are deployed in the cluster, each node is configured with *N* disks, the size of each disk is *M*, the cluster contains *n* topics (named as T1, T2, ..., Tn), the data input traffic per second of the *m* topic is *X* (Tm) MB/s, the number of configured replicas is *R* (Tm), and the configured data retention time is *Y* (Tm) hour, the following requirement must be met:

$$M \times N \times K > \sum_{i=T1}^{Tn} (X(i)R(i)Y(i) \times 3600)$$

- If the size of a disk is *M*, the disk has *n* partitions (named as P0, P1, ..., Pn), the data write traffic per second of the *m* partition is *Q* (Pm) MB/s (calculation method: data traffic of the topic to which the *m* partition belongs divided by the number of partitions), and the data retention time is *T* (Pm) hours, the following requirement must be met for the disk:

$$M > \sum_{i=P0}^{Pn} (Q(i)T(i) \times 3600)$$

- Based on the throughput, if the throughput that can be reached by the producer is *P*, the throughput that can be reached by the consumer is *C*, and the expected throughput of Kafka is *T*, it is recommended that the number of partitions of the topic be set to Max(T/P, T/C).

NOTE

- In a Kafka cluster, more partitions mean higher throughput. However, too many partitions also pose potential impacts, such as a file handle increase, unavailability increase (for example, if a node is faulty, the time window becomes large after the leader is reselected in some partitions), and end-to-end latency increase.
- Suggestion: The disk usage of a partition is smaller than or equal to 100 GB; the number of partitions on a node is smaller than or equal to 3,000; the number of partitions in the entire cluster is smaller than or equal to 10,000.

18.10.2 Kafka Feature Description

Kafka Idempotent Feature

Feature description: The function of creating idempotent producers is introduced in Kafka 0.11.0.0. After this function is enabled, producers are automatically upgraded to idempotent producers. When producers send messages with the same field values, brokers automatically detect whether the messages are duplicate to avoid duplicate data. Note that this feature can only ensure idempotence in a single partition. That is, an idempotent producer can ensure that no duplicate messages exist in a partition of a topic. Only idempotence on a single session can be implemented. The session refers to the running of the producer process. That is, idempotence cannot be ensured after the producer process is restarted.

Method for enabling this feature:

1. Add **props.put("enable.idempotence", true)** to the secondary development code.
2. Add **enable.idempotence = true** to the client configuration file.

Kafka Transaction Feature

Feature description: Kafka 0.11 introduces the transaction feature. The Kafka transaction feature indicates that a series of producer message production and consumer offset submission operations are in the same transaction, or are regarded as an atomic operation. Message production and offset submission succeed or fail at the same time. This feature provides transactions at the Read Committed isolation level to ensure that multiple messages are written to the target partition atomically and that the consumer can view only the transaction messages that are successfully submitted. The transaction feature of Kafka is used in the following scenarios:

1. Multiple pieces of data sent by a producer can be encapsulated in a transaction to form an atomic operation. All messages are successfully sent or fail to be sent.

2. read-process-write mode: Message consumption and production are encapsulated in a transaction to form an atomic operation. In a streaming application, a service usually needs to receive messages from the upstream system, process the messages, and then send the processed messages to the downstream system. This corresponds to message consumption and production.

Example of secondary development code:

```
// Initialize the configuration and enable the transaction feature.
Properties props = new Properties();
props.put("enable.idempotence", true);
props.put("transactional.id", "transaction1");
...

KafkaProducer producer = new KafkaProducer<String, String>(props);

// init transaction
producer.initTransactions();
try {
    // Start a transaction.
    producer.beginTransaction();
    producer.send(record1);
    producer.send(record2);
    // Stop a transaction.
    producer.commitTransaction();
} catch (KafkaException e) {
    // Abort a transaction.
    producer.abortTransaction();
}
```

Nearby Consumption

Feature description: In versions earlier than Kafka 2.4.0, the production and consumption of the client are leader copies oriented to each partition. Follower copies are used only for data redundancy and do not provide services for external systems. As a result, the leader copy has high pressure. In addition, in cross-DC and cross-rack consumption scenarios, a large volume of data is transmitted between DCs and between racks. In Kafka 2.4.0 and later versions, the Kafka kernel can consume data from follower replicas, which greatly reduces the data transmission volume and reduces the network bandwidth pressure in cross-DC and cross-rack scenarios. The community opens the ReplicaSelector API to support this feature. By default, MRS Kafka provides two methods to use this API.

1. **RackAwareReplicaSelector**: indicates that replicas in the same rack are preferentially consumed (nearby consumption in a rack).
2. **AzAwareReplicaSelector**: indicates that copies from nodes in the same AZ are preferentially consumed (nearby consumption in an AZ).

The following uses **RackAwareReplicaSelector** as an example to describe how to consume the closest replica.

```
public class RackAwareReplicaSelector implements ReplicaSelector {

  @Override
  public Optional<ReplicaView> select(TopicPartition topicPartition,
                                      ClientMetadata clientMetadata,
                                      PartitionView partitionView) {
    if (clientMetadata.rackId() != null && !clientMetadata.rackId().isEmpty()) {
      Set<ReplicaView> sameRackReplicas = partitionView.replicas().stream()
              // Filter the replicas that are in the same rack as the client.
              .filter(replicaInfo -> clientMetadata.rackId().equals(replicaInfo.endpoint().rack()))
              .collect(Collectors.toSet());
```



```
            if (sameRackReplicas.isEmpty()) {
                // If no replicas are in the same rack as the client, the leader replica is returned.
                return Optional.of(partitionView.leader());
            } else {
                // It shows that a replica that is in the same rack as the client exists.
                if (sameRackReplicas.contains(partitionView.leader())) {
                    // If the client and the leader replica are in the same rack, the leader replica returns first.
                    return Optional.of(partitionView.leader());
                } else {
                    // Otherwise, the latest replica synchronized with the leader is returned.
                    return sameRackReplicas.stream().max(ReplicaView.comparator());
                }
            }
        } else {
            // If the rack information is not contained in the client request, the leader replica is returned first.
            return Optional.of(partitionView.leader());
        }
    }
}
```

Method for enabling this feature:

1. Server: Update the **replica.selector.class** configuration item based on different features.
   - To enable "nearby consumption in a rack", set this parameter to **org.apache.kafka.common.replica.RackAwareReplicaSelector**.
   - To enable "nearby consumption in an AZ", set this parameter to **org.apache.kafka.common.replica.AzAwareReplicaSelector**.
2. Client: Add the **client.rack** configuration item to the **consumer.properties** file in the *{Client installation directory}*/**Kafka/kafka/config** directory.
   - If the "nearby consumption in a rack" is enabled on the server, add the information about the rack where the client is located, for example, **client.rack = /default0/rack1**.
   - If the "nearby consumption in an AZ" is enabled on the server, add the information about the rack where the client is located, for example, **client.rack = /AZ1/rack1**.

Ranger Unified Authentication

Feature description: In versions earlier than Kafka 2.4.0, Kafka supports only the SimpleAclAuthorizer authentication plugin provided by the community. In Kafka 2.4.0 and later versions, MRS Kafka supports both the Ranger authentication plugin and the authentication plugin provided by the community. Ranger authentication is used by default. Based on the Ranger authentication plugin, fine-grained Kafka ACL management can be performed.

NOTE

If the Ranger authentication plugin is used on the server and **allow.everyone.if.no.acl.found** is set to **true**, all actions are allowed when a non-secure port is used for access. You are advised to disable **allow.everyone.if.no.acl.found** for security clusters that use the Ranger authentication plugin.

18.10.3 Synchronizing Binlog-based MySQL Data to the MRS Cluster

This section describes how to use the Maxwell data synchronization tool to migrate offline binlog-based data to an MRS Kafka cluster.

Maxwell is an open-source application that reads MySQL binlogs, converts operations, such as addition, deletion, and modification, into a JSON format, and sends them to an output end, such as a console, a file, and Kafka. For details about Maxwell, visit **https://maxwells-daemon.io**. Maxwell can be deployed on a MySQL server or on other servers that can communicate with MySQL.

Maxwell runs on a Linux server, including EulerOS, Ubuntu, Debian, CentOS, and OpenSUSE. Java 1.8+ must be supported.

The following provides details about data synchronization.

1. **Configuring MySQL**
2. **Installing Maxwell**
3. **Configuring Maxwell**
4. **Starting Maxwell**
5. **Verifying Maxwell**
6. **Stopping Maxwell**
7. **Format of the Maxwell Generated Data and Description of Common Fields**

Configuring MySQL

Step 1 Start the binlog, open the **my.cnf** file in MySQL, and check whether **server_id**, **log-bin**, and **binlog_format** are configured in the **[mysqld]** block. If they are not configured, run the following command to add configuration items and restart MySQL. If they are configured, skip this step.

```
$ vi my.cnf

[mysqld]
server_id=1
log-bin=master
binlog_format=row
```

Step 2 Maxwell needs to connect to MySQL, create a database named **maxwell** for storing metadata, and access the database to be synchronized. Therefore, you are advised to create a MySQL user for Maxwell to use. Log in to MySQL as user **root** and run the following commands to create a user named **maxwell** (*XXXXXX* indicates the password and needs to be replaced with actual one). Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

- If Maxwell is deployed on a non-MySQL server, the created user **maxwell** must have a permission to remotely log in to the database. In this case, run the following command to create the user:

  mysql> GRANT ALL on maxwell.* to 'maxwell'@'%' identified by 'XXXXXX';

  mysql> GRANT SELECT, REPLICATION CLIENT, REPLICATION SLAVE on *.* to 'maxwell'@'%';

- If Maxwell is deployed on the MySQL server, the created user **maxwell** can be configured to log in to the database only on the local host. In this case, run the following command:

  mysql> GRANT SELECT, REPLICATION CLIENT, REPLICATION SLAVE on *.* to 'maxwell'@'localhost' identified by 'XXXXXX';

mysql> GRANT ALL on maxwell.* to 'maxwell'@'localhost';

----End

Installing Maxwell

Step 1 Download the installation package at **https://github.com/zendesk/maxwell/releases** and select the **maxwell-XXX.tar.gz** binary file for download. In the file name, **XXX** indicates a version number.

Step 2 Upload the **tar.gz** package to any directory (the **/opt** directory of the Master node used as an example here).

Step 3 Log in to the server where Maxwell is deployed and run the following command to go to the directory where the **tar.gz** package is stored.

cd /opt

Step 4 Run the following commands to decompress the **maxwell-XXX.tar.gz** package and go to the **maxwell-XXX** directory:

tar -zxvf maxwell-XXX.tar.gz

cd maxwell-XXX

----End

Configuring Maxwell

If the **conf** directory exists in the **maxwell-XXX** folder, configure the **config.properties** file. For details about the configuration items, see **Table 18-20**. If the **conf** directory does not exist, change **config.properties.example** in the **maxwell-XXX** folder to **config.properties**.

Table 18-20 Maxwell configuration item description

Parameter	Mandatory	Description	Default Value
user	Yes	Name of the user for connecting to MySQL, that is, the user created in **Step 2**.	-
password	Yes	Password for connecting to MySQL. There can be security risks if a configuration file contains the authentication password. You are advised to delete the configuration file or use other secure methods to keep the password.	-
host	No	MySQL address	localhost
port	No	MySQL port	3306

Parameter	Mandatory	Description	Default Value
log_level	No	Log print level. The options are as follows: • debug • info • warn • error	info
output_ddl	No	Whether to send a DDL (modified based on definitions of the database and data table) event • **true**: Send DDL events. • **false**: Do not send DDL events.	false
producer	Yes	Producer type. Set this parameter to **kafka**. • **stdout**: Log the generated events. • **kafka**: Send the generated events to Kafka.	stdout
producer_partition_by	No	Partition policy used to ensure that data of the same type is written to the same partition of Kafka. • **database**: Events of the same database are written to the same partition of Kafka. • **table**: Events of the same table are written to the same partition of Kafka.	database
ignore_producer_error	No	Specifies whether to ignore the error that the producer fails to send data. • **true**: The error information is logged and the error data is skipped. The program continues to run. • **false**: The error information is logged and the program is terminated.	true
metrics_slf4j_interval	No	Interval for outputting statistics on data successfully uploaded or failed to be uploaded to Kafka in logs. The unit is second.	60
kafka.bootstrap.servers	Yes	Address of the Kafka proxy node. The value is in the format of **HOST:PORT[,HOST:PORT]**.	-
kafka_topic	No	Name of the topic that is written to Kafka	maxwell
dead_letter_topic	No	Kafka topic used to record the primary key of the error log record when an error occurs when the record is sent	-

Parameter	Mandatory	Description	Default Value
kafka_version	No	Kafka producer version used by Maxwell, which cannot be configured in the **config.properties** file. You need to use the **--kafka_version xxx** parameter to import the version number when starting the command.	-
kafka_partition_hash	No	Kafka topic partitioning algorithm. The value can be **default** or **murmur3**.	default
kafka_key_format	No	Key generation method of the Kafka record. The value can be **array** or **Hash**.	Hash
ddl_kafka_topic	No	Topic that is written to the DDL operation when **output_ddl** is set to **true**	{kafka_topic}
filter	No	Used to filter databases or tables. ● If only the **mydatabase** database needs to be collected, set this parameter to the following: exclude: *.*,include: mydatabase.* ● If only the **mydatabase.mytable** table needs to be collected, set this parameter to the following: exclude: *.*,include: mydatabase.mytable ● If only the **mytable**, **mydate_123**, and **mydate_456** tables in the **mydatabase** database need to be collected, set this parameter to the following: exclude: *.*,include: mydatabase.mytable,include: mydatabase./mydate_\\d*/	-

Starting Maxwell

Step 1 Log in to the server where Maxwell is deployed.

Step 2 Run the following command to go to the Maxwell installation directory:

cd /opt/maxwell-1.21.0/

NOTE

For the first time to use Maxwell, you are advised to change **log_level** in **conf/config.properties** to **debug** (debug level) so that you can check whether data can be obtained from MySQL and sent to Kafka after startup. After the entire process is debugged, change **log_level** to **info**, and then restart Maxwell for the modification to take effect.

```
# log level [debug | info | warn | error]
log_level=debug
```

Step 3 Run the following commands to start Maxwell:

source /opt/client/bigdata_env

bin/Maxwell

**bin/maxwell --user='maxwell' --password='XXXXXX' --host='127.0.0.1' **

--producer=kafka --kafka.bootstrap.servers=kafkahost:9092 --kafka_topic=Maxwell

In the preceding commands, **user**, **password**, and **host** indicate the username, password, and IP address of MySQL, respectively. You can configure the three parameters by modifying configurations of the configuration items or using the preceding commands. **kafkahost** indicates the IP address of the Core node in the streaming cluster.

If information similar to the following appears, Maxwell has started successfully:

```
Success to start Maxwell [78092].
```

----End

Verifying Maxwell

Step 1 Log in to the server where Maxwell is deployed.

Step 2 View the logs. If the log file does not contain an ERROR log and the following information is displayed, the connection between Maxwell and MySQL is normal:

```
BinlogConnectorLifecycleListener - Binlog connected.
```

Step 3 Log in to the MySQL database and update, create, or delete test data. The following provides operation statement examples for your reference.

```
--Creating a database
create database test;
--Creating a table
create table test.e (
  id int(10) not null primary key auto_increment,
  m double,
  c timestamp(6),
  comment varchar(255) charset 'latin1'
);
-- Adding a record
insert into test.e set m = 4.2341, c = now(3), comment = 'I am a creature of light.';
--Updating a record
update test.e set m = 5.444, c = now(3) where id = 1;
--Deleting a record
delete from test.e where id = 1;
--Modifying a table
alter table test.e add column torvalds bigint unsigned after m;
--Deleting a table
drop table test.e;
-- Deleting a database
drop database test;
```

Step 4 Check the Maxwell logs. If no WARN/ERROR is displayed, Maxwell is installed and configured properly.

To check whether the data is successfully uploaded, set **log_level** in the **config.properties** file to **debug**. When the data is successfully uploaded, the following JSON data is printed immediately. For details about the fields, see **Format of the Maxwell Generated Data and Description of Common Fields**.

```
{"database":"test","table":"e","type":"insert","ts":1541150929,"xid":60556,"commit":true,"data":
{"id":1,"m":4.2341,"c":"2018-11-02 09:28:49.297000","comment":"I am a creature of light."}}
......
```


NOTE

After the entire process is debugged, you can change the value of **log_level** in the **config.properties** file to **info** to reduce the number of logs to be printed and restart Maxwell for the modification to take effect.

```
# log level [debug | info | warn | error]
log_level=info
```

----End

Stopping Maxwell

Step 1 Log in to the server where Maxwell is deployed.

Step 2 Run the command to obtain the Maxwell process ID (PID). The second field in the command output is PID.

ps -ef | grep Maxwell | grep -v grep

Step 3 Run the following command to forcibly stop the Maxwell process:

kill -9 PID

----End

Format of the Maxwell Generated Data and Description of Common Fields

The data generated by Maxwell is in JSON format. The common fields are described as follows:

- **type**: operation type. The options are **database-create**, **database-drop**, **table-create**, **table-drop**, **table-alter**, **insert**, **update**, and **delete**.
- **database**: name of the database to be operated
- **ts**: operation time, which is a 13-digit timestamp
- **table**: name of the table to be operated
- **data**: content after data is added, deleted, or modified
- **old**: content before data is modified or schema definition before a table is modified
- **sql**: SQL statement for DDL operations
- **def**: schema definition for table creation and modification
- **xid**: unique ID of an object
- **commit**: check whether such operations as data addition, deletion, and modification have been submitted.

18.10.4 How Do I Solve the Problem that Kafka Topics Cannot Be Deleted?

Question

How do I delete a Kafka topic if it fails to be deleted?

Answer

- Possible cause 1: The **delete.topic.enable** configuration item is not set to **true**. The deletion can be performed only when the configuration item is set to **true**.
- Possible cause 2: The **auto.create.topics.enable** configuration parameter is set to **true**, which is used by other applications and is always running in the background.

Solution:

- For cause 1: Set **delete.topic.enable** to **true** on the configuration page.
- For cause 2: Stop the application that uses the topic in the background, or set **auto.create.topics.enable** to **false** (restart the Kafka service), and then delete the topic.

19 Using Kudu

19.1 Using Kudu from Scratch

Kudu is a columnar storage manager developed for the Apache Hadoop platform. Kudu shares the common technical properties of Hadoop ecosystem applications. It is horizontally scalable and supports highly available operations.

Prerequisites

The cluster client has been installed. For example, the client is installed in the **/opt/hadoopclient** directory. The client directory in the following operations is only an example. Change it to the actual installation directory.

Procedure

Step 1 Log in to the node where the client is installed as the client installation user.

Run the **su - omm** command to switch to user **omm**.

Step 2 Run the following command to go to the client installation directory:

cd /opt/hadoopclient

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Run the Kudu command line tool.

Run the command line tool of the Kudu component to view help information.

kudu -h

The command output is as follows:

```
Usage: kudu <command> [<args>]

<command> can be one of the following:
       cluster   Operate on a Kudu cluster
      diagnose   Diagnostic tools for Kudu servers and clusters
            fs   Operate on a local Kudu filesystem
          hms   Operate on remote Hive Metastores
```



```
 local_replica   Operate on local tablet replicas via the local filesystem
        master   Operate on a Kudu Master
           pbc   Operate on PBC (protobuf container) files
          perf   Measure the performance of a Kudu cluster
remote_replica   Operate on remote tablet replicas on a Kudu Tablet Server
         table   Operate on Kudu tables
        tablet   Operate on remote Kudu tablets
          test   Various test actions
       tserver   Operate on a Kudu Tablet Server
           wal   Operate on WAL (write-ahead log) files
```

NOTE

The Kudu command line tool does not support DDL and DML operations, but provides the refined query function for the **cluster**, **master**, **tserver**, **fs**, and **table** parameters.

Common operations:

- Check the tables in the current cluster.

  kudu table list *KuduMaster instance IP1:7051, KuduMaster instance IP2:7051, KuduMaster instance IP3:7051*

- Query the configurations of the KuduMaster instance of the Kudu service.

  kudu master get_flags *KuduMaster instance IP:7051*

- Query the schema of a table.

  kudu table describe *KuduMaster instance IP1:7051, KuduMaster instance IP2:7051, KuduMaster instance IP3:7051 Table name*

- Delete a table.

  kudu table delete *KuduMaster instance IP1:7051, KuduMaster instance IP2:7051, KuduMaster instance IP3:7051 Table name*

  NOTE

  To obtain the IP address of the KuduMaster instance, choose **Components** > **Kudu** > **Instances** on the cluster details page.

----End

19.2 Accessing the Kudu Web UI

You can view Kudu job information on the Kudu web UI.

Prerequisites

Kudu has been installed in a cluster.

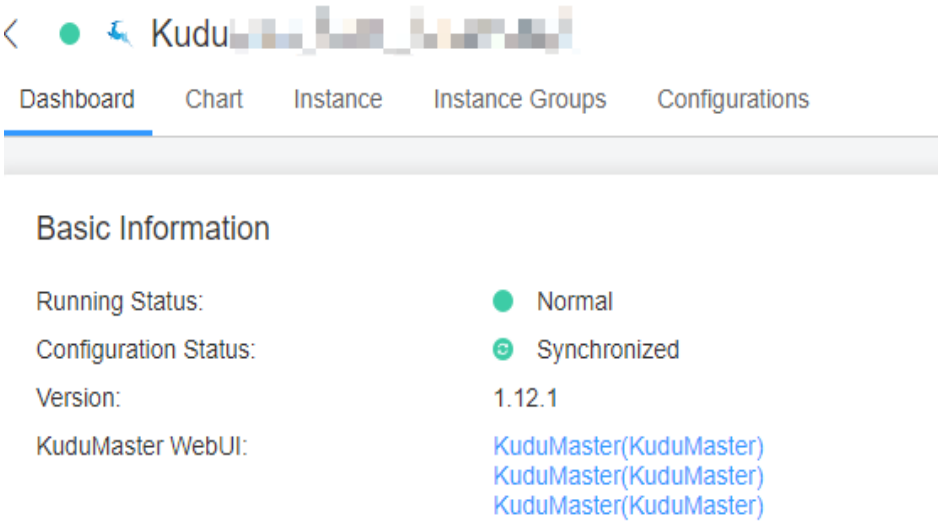
Accessing KuduMaster WebUI (MRS 3.x or Later)

Step 1 Log in to Manager. For details, see **Accessing FusionInsight Manager**.

Step 2 Choose **Cluster** > **Services**> **Spark**.

Step 3 In the **Dashboard** page of Kudu, click **KuduMaster(KuduMaster)** on the right side of **KuduMaster WebUI**. The KuduMaster web UI is displayed.

Figure 19-1 KuduMaster WebUI



----End

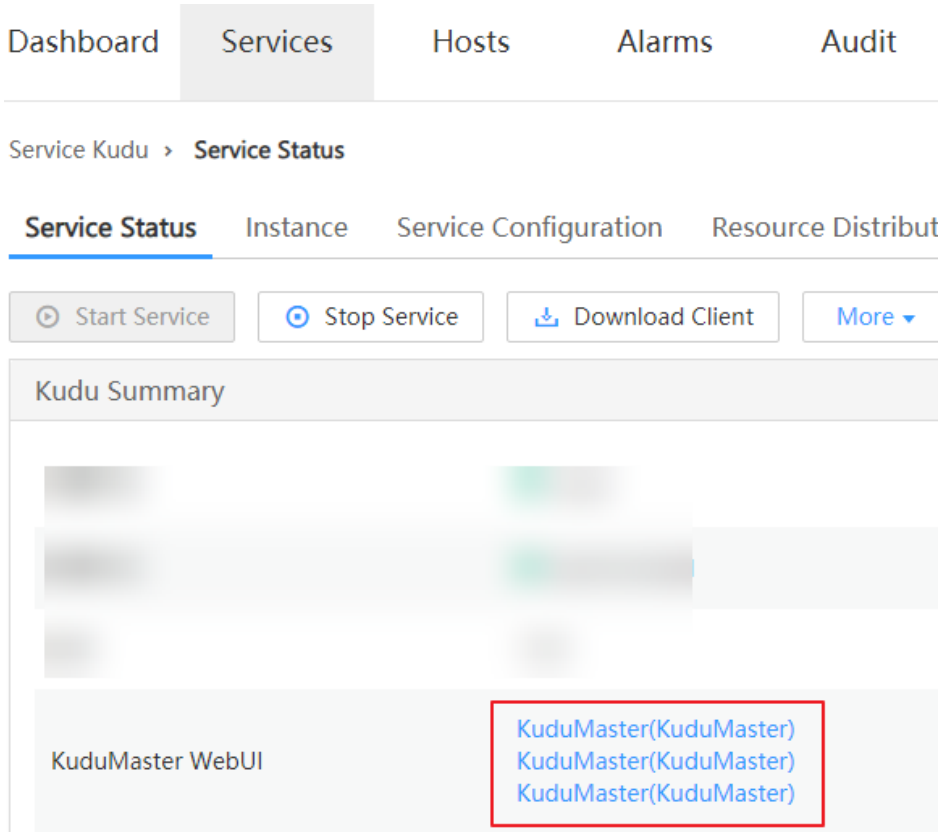
Accessing KuduMaster WebUI (Versions Earlier Than MRS 3.*x*)

Step 1 Access Manager. For details, see **Accessing FusionInsight Manager**.

Step 2 choose **Services** > **Kudu**.

Step 3 In **KuduMaster WebUI** of **Kudu Summary**, click **KuduMaster(KuduMaster)**. The KuduMaster web UI is displayed.

Figure 19-2 KuduMaster WebUI



----End

20 Using Loader

20.1 Overview of Importing and Exporting Loader Data

Introduction to Loader Data Import

Loader is an ETL tool that enables MRS to exchange data and files with external data sources, such as relational databases, SFTP servers, and FTP servers. It allows data or files to be imported from relational databases or file systems to MRS.

Loader supports the following data import modes:

- Importing data from a relational database to HDFS or OBS
- Importing data from a relational database to HBase
- Importing data from a relational database to Phoenix tables
- Importing data from a relational database to Hive tables
- Importing data from an SFTP server to HDFS or OBS
- Importing data from an SFTP server to HBase
- Importing data from an SFTP server to Phoenix tables
- Importing data from an SFTP server to Hive tables
- Importing data from an FTP server to HDFS or OBS
- Importing data from an FTP server to HBase
- Importing data from an FTP server to Phoenix tables
- Importing data from an FTP server to Hive tables
- Importing data from HDFS or OBS to HBase in the same cluster

MRS needs to connect to an external data source to exchange data and files with the data source. The following connectors are used to configure connection parameters for different types of data sources

- generic-jdbc-connector: relational database connector
- ftp-connector: FTP data source connector
- hdfs-connector: HDFS data source connector

- oracle-connector: dedicated connector for Oracle databases. **row_id** serves as partition columns. Compared with generic-jdbc-connector, Map jobs are more evenly distributed on oracle-connector, and whether indexes have been created for the partition columns does not matter.
- mysql-fastpath-connector: dedicated connector for MySQL databases. Data is imported and exported by using the mysqldump and mysqlimport tools of MySQL. Compared with generic-jdbc-connector, mysql-fastpath-connector delivers a faster data import and export speed.
- sftp-connector: SFTP data source connector
- oracle-partition-connector: connector that supports the Oracle partition feature, which is used to optimize the import and export of Oracle partition tables.

NOTE

- When an FTP data source connector is used, data is not encrypted. This may result in security risks. You are advised to use an SFTP data source connector.
- You are advised to deploy the SFTP server, FTP server, database server, and Loader into separate subnets to ensure secure data import.
- For connection to relational databases, general database connectors (generic-jdbc-connector) or dedicated database connectors (oracle-connector, oracle-partition-connector, and mysql-fastpath-connector) are available. However, compared with general database connectors, dedicated database connectors perform better in data import and export because they are optimized for specific database types.
- When mysql-fastpath-connector is used, the **mysqldump** and **mysqlimport** commands of MySQL must be available on NodeManager nodes, and the MySQL client version to which the two commands belong must be compatible with the MySQL server version. If the two commands are unavailable or the versions are incompatible, install the MySQL client applications and tools following the instructions at **http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**.
- When oracle-connector is used, the connection user must be granted the select permission on the following system catalogs or views:

  dba_tab_partitions, dba_constraints, dba_tables, dba_segments, v$version, dba_objects, v$instance, SYS_CONTEXT, dba_extents, and dba_tab_subpartitions
- When oracle-partition-connector is used, the connection user must be granted the select permission on the following system catalogs: dba_objects and dba_extents.

Introduction to Loader Data Export

Loader is an extract, transform, and load (ETL) tool for exchanging data and files between MRS and relational databases and file systems. You can use the Loader to export data or files from MRS to the relational databases or file systems.

Loader supports the following data export modes:

- Exporting data from HDFS or OBS to an SFTP server
- Exporting data from HDFS or OBS to a relational database
- Exporting data from HBase to an SFTP server
- Exporting data from HBase to a relational database
- Exporting data from Phoenix tables to an SFTP server
- Exporting data from Phoenix tables to a relational database
- Exporting data from Hive to an SFTP server

- Exporting Data from Hive to a Relational Database
- Exporting data from HBase to HDFS or OBS in the same cluster

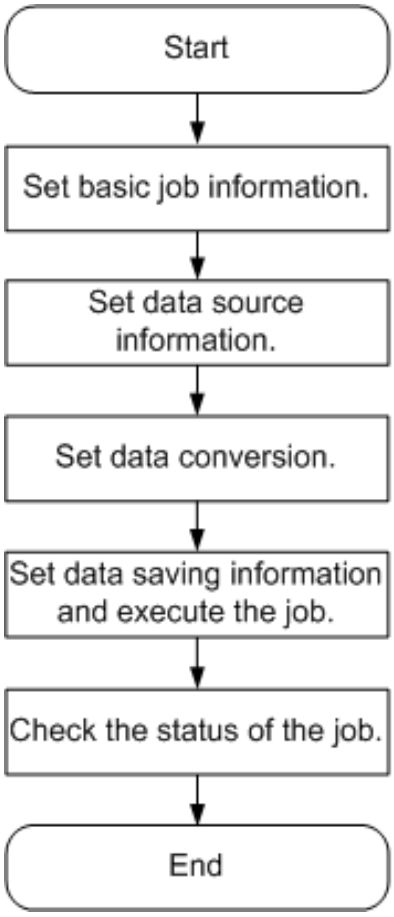
The MRS needs to connect to the data source to exchange data and files with the external data source. The following connectors are used to configure connection parameters for different types of data sources:

- **generic-jdbc-connector**: relational database connector
- **hdfs-connector**: HDFS data source connector
- **oracle-connector**: specifies a dedicated connector for Oracle databases. **row_id** serves as partition columns. Compared with generic-jdbc-connector, Map tasks are more evenly distributed on oracle-connector, and whether the partition columns have created indexes does not affect oracle-connector.
- **mysql-fastpath-connector**: specifies a dedicated connector for MySQL databases. Data is imported and exported by using the mysqldump and mysqlimport tools of MySQL. Compared with generic-jdbc-connector, the data import and export speed is faster.
- sftp-connector: SFTP data source connector
- oracle-partition-connector: connector that supports the Oracle partition feature, which is used to optimize the import and export of Oracle partition tables.

Import Process

You can import and export data on the Loader page. **Figure 20-1** shows the data import process.

Figure 20-1 Import process



You can also use shell scripts to update and run Loader jobs. To use this method, you need to configure the installed Loader client.

20.2 Loader User Permission Management

20.2.1 Creating a Loader Role

Scenario

Create and configure a Loader role on FusionInsight Manager as an MRS cluster administrator. The Loader role can set Loader administrator permissions, job connections, job groups, and Loader job operation and scheduling permissions.

Prerequisites

- The MRS cluster administrator has understood service requirements.
- You have logged in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.

Procedure

Step 1 Choose **System** > **Permission** > **Role**.

Step 2 Click **Create Role** and set a role name and enter description.

Step 3 Set permissions. For details, see **Table 20-1**.

NOTE

When setting permissions for a role, you cannot set permissions for multiple resources at the same time. If you need to set permissions for multiple resources, set them one by one.

Loader permissions:

- **Admin**: Loader administrator permission
- **Job Connector**: connection permission of Loader
- **Job Group**: permission to perform operations on Loader job groups. You can set the operation permissions of a specific job in a specified job group, including the **Edit** and **Execute** permissions of the job.
- **Job Scheduler**: permission to schedule Loader jobs

Table 20-1 Setting Loader roles

Task	Role Authorization
Setting the Loader administrator permission	In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Loader** and select **Admin**.
Setting the connection permission of Loader (including the editing, deletion, and reference permissions of Job Connection)	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Loader** > **Job Connector**. 2. In the **Permission** column of the specified job link, select **Edit**.
Setting the edit permission for Loader job groups (including modifying the name of a job group, deleting a specified group, creating jobs in a specified group, importing jobs from external systems to a specified group in batches, and migrating jobs from other groups to a specified group)	1. In the **Permission** table, choose **Loader** > **Job Group**. 2. In the **Permission** column of the specified job group, select **Edit Group**.

Task	Role Authorization
Setting the edit permission for all jobs in a Loader job group (including the permission to edit all existing or new jobs in the group)	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Loader** > **Job Group**. 2. In the **Permission** column of the specified job group, select **Edit Job**.
Setting the execution permission for all jobs in the Loader job group (including the execution permission on all existing or new jobs in the group)	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Loader** > **Job Group**. 2. In the **Permission** column of the specified job group, select **Execute Job**.
Setting the edit permission for Loader jobs (including editing, deleting, copying, and exporting jobs)	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Loader** > **Job Group**. 2. Select a job group. 3. In the **Permission** column of the specified job, select **Edit**.
Setting the execution permission for Loader jobs (including permissions to start and stop jobs and view historical records)	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Loader** > **Job Group**. 2. Select a job group. 3. In the **Permission** column of the specified job, select **Execute**.
Setting the permission for scheduling Loader jobs (including the editing, deletion, and validation permissions of Scheduler)	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Loader** > **Job Scheduling**. 2. In the **Permission** column of the specified job scheduling row, select **Edit**.

NOTE

1. In addition to **Admin**, the preceding permissions are configured only for inventory resource information.
2. Users without the preceding roles can also create tasks, groups, and connectors, but cannot perform operations on inventory resources.

Step 4 Click **OK**, and return to the **Role** page.

----End

20.3 Uploading the MySQL Database Connection Driver

Scenario

As a component for batch data export, Loader can import and export data using a relational database. Before connecting to a relational database, you need to manually upload the driver.

Procedure

Modify the permission on the JAR package of the relational database driver.

Step 1 Log in to the active and standby management nodes of the Loader service, obtain the driver JAR package of the relational database, and save it to the following directory on the active and standby Loader nodes: **${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib**

Step 2 Run the following commands as user **root** on the active and standby nodes of the Loader service to change the permission:

cd ${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

chown omm:wheel *JAR package name*

chmod 600 *JARpackage name*

Step 3 Log in to FusionInsight Manager. Choose **Cluster** and click the target cluster name. In the navigation pane on the left, choose **Services** > **Loader**. In the upper right corner, choose **More**, select **Restart Service**, and enter the password of the administrator to restart the Loader service.

----End

20.4 Creating a Loader Data Import Job

20.4.1 Using Loader to Import Data to an MRS Cluster

Scenario

This section describes how to import data from external data sources to MRS.

Generally, you can manually manage data import and export jobs on the Loader UI. To use shell scripts to update and run Loader jobs, you must configure the installed Loader client.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have the permission to access the HDFS or OBS directories, HBase tables, and data involved in job execution.
- You have obtained the username and password used by an external data source (SFTP server or relational database).
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- When using Loader to import data from SFTP, FTP, and HDFS/OBS, ensure that the input paths and input path subdirectories of the external data sources and the name of the files in these directories do not contain any of the special characters /"';:;.
- If a task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.

Procedure

Step 1 Check whether data is imported from MRS to a relational database for the first time.

- If yes, go to **Step 2**.
- If no, go to **Step 3**.

Step 2 Modify the permission on the JAR package of the relational database driver.

1. Log in to the active and standby management nodes of the Loader service, obtain the JAR package of the relational database driver, and save it to the following directory on the active and standby Loader nodes: **${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib**.
2. Run the following command as user **root** on the active and standby nodes of the Loader service to modify the permission:

   cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

   chown omm:wheel *JAR package name*

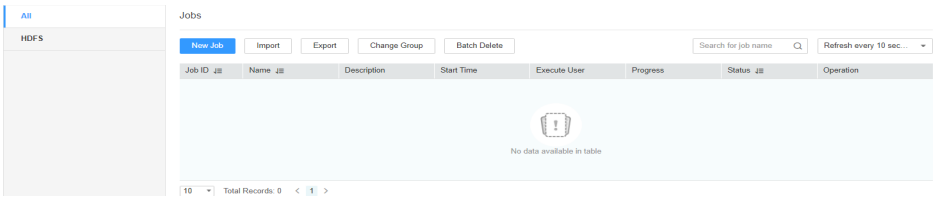
   chmod 600 *JAR package name*
3. Log in to FusionInsight Manager. Choose **Cluster** > **Services** > **Loader** > **More** > **Restart Service**. Enter the password of the administrator to restart the Loader service.

Step 3 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.

3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-2 Loader web UI



Step 4 Create a Loader data import job. Click **New Job**. Select the required job type in **1. Basic Information**, and click **Next**.

1. Set **Name** to the job name and **Type** to **Import**.
2. Select a connection for **Connection**. By default, no connection is created. Click **Add** to create a connection, and then click **Test** to test whether the connection is available. Click **OK** when the system displays a message indicates that the test is successful.

   Data sources need to be connected when MRS exchanges data and files with external data sources. **Connection** indicates the set of connection parameters for connecting to data sources.

Table 20-2 Connection configuration parameters

Connector	Parameter	Description
generic-jdbc-connector	JDBC Driver Class	Name of a JDBC driver class
	JDBC Connection String	JDBC connection string
	Username	Username for connecting to the database
	Password	Password for connecting to the database
	JDBC Connection Properties	JDBC connection attribute. Click **Add** to manually add connection attributes. – **Name**: connection attribute name – **Value**: connection attribute value
ftp-connector	FTP Server IP Address	IP address of the FTP server
	FTP Server Port	Port number of the FTP server
	FTP Username	Username for accessing the FTP server
	FTP Password	Password for accessing the FTP server

Connector	Parameter	Description
	FTP Mode	FTP access mode. Possible values are **ACTIVE** and **PASSIVE**. If this parameter is not set, FTP access is in passive mode by default.
	FTP Protocol	FTP protocol. – **FTP**: indicates the FTP protocol. – **SSL_EXPLICIT**: indicates the explicit SSL protocol. – **SSL_IMPLICIT**: indicates the implicit SSL protocol. – **TLS_EXPLICIT**: indicates the explicit TLS protocol. – **TLS_IMPLICIT**: indicates the implicit TLS protocol. If this parameter is not set, the FTP protocol is used by default.
	File Name Encoding Type	File name and file path encoding format supported by the FTP server. If this parameter is not set, the default format UTF-8 is used.
hdfs-connector	-	-
oracle-connector	JDBC Connection String	Connection string for a user to connect to the database
	Username	Username for connecting to the database
	Password	Password for connecting to the database
	Connection Properties	Connection attributes. Click **Add** to manually add connection attributes. – **Name**: connection attribute name – **Value**: connection attribute value
mysql-fastpath-connector	JDBC Connection String	JDBC connection string
	Username	Username for connecting to the database
	Password	Password for connecting to the database
	Connection Properties	Connection attributes. Click **Add** to manually add connection attributes. – **Name**: connection attribute name – **Value**: connection attribute value

Connector	Parameter	Description
sftp-connector	SFTP Server IP Address	IP address of the SFTP server
	SFTP Server Port	Port number of the SFTP server
	SFTP Username	Username for accessing the SFTP server
	SFTP Password	Password for accessing the SFTP server
	SFTP Public Key	Public key of the SFTP server
oracle-partition-connector	JDBC Driver Class	Name of a JDBC driver class
	JDBC Connection String	JDBC connection string
	Username	Username for connecting to the database
	Password	Password for connecting to the database
	Connection Properties	Connection attributes. Click **Add** to manually add connection attributes. – **Name**: connection attribute name – **Value**: connection attribute value

3. Set **Group** to the group to which the job belongs. By default, there is no created group. Click **Add** to create a group and click **OK**.
4. **Queue** indicates that Loader tasks are executed in a specified Yarn queue. The default value is **root.default**, which indicates that the tasks are executed in the **default** queue.
5. Set **Priority** to the priority of Loader tasks in the specified Yarn queue. The value can be **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, or **VERY_HIGH**. The default value is **NORMAL**.

Step 5 In the **2. Input Settings** area, set the data source and click **Next**.

NOTE

- When creating or editing a Loader job, you can use macro definitions when configuring parameters such as the SFTP path, HDFS/OBS path, and Where condition of SQL. For details, see **Using Macro Definitions in Configuration Items** .
- Loader supports common field data types, such as Char, VarChar, Boolean, Binary, SmallInt, Int, BigInt, Decimal, Float, Double, Date, Time, TimeStamp, and String. The supported types may vary according to the data source. For details about the supported types, expand the field data type drop-down list of the corresponding input operator (such as Table Input) on the Loader GUI. Some database-specific fields may not be supported. For example, Loader does not support the CLOB, XMLType, and BLOB fields in Oracle.

Table 20-3 List of input configuration parameters

Source File Type	Parameter	Description
sftp-connector or ftp-connector	Input Path	Input path or name of the source file on an SFTP server. If multiple SFTP server IP addresses are configured for the connector, you can set this parameter to multiple input paths separated with semicolons (;). Ensure that the number of input paths is the same as that of SFTP servers configured for the connector.
	File Split Type	Indicates whether to split source files by file name or size. The files obtained after the splitting are used as the input files of each Map in the MapReduce task for data import. **FILE** indicates that each Map processes one or more complete source files. The same source file cannot be allocated to different Maps. When the data is saved to the output directory, the directory structure of the input path is retained. **SIZE** indicates that each Map processes input files of a certain size. A source file can be split into multiple Maps. The number of files saved when data is saved to the output directory is the same as that of Maps. The file name format is **import_part**_*xxxx*, where *xxxx* is a unique random number generated by the system.
	Filter Type	File filtering criterion. **WILCARD** indicates that a wildcard is used in filtering, and **REGEX** indicates that a regular expression is used in filtering. This parameter is used together with **Path Filter** and **File Filter**. The default value is **WILDCARD**.
	Path Filter	Wildcard or regular expression for filtering the directories in the input path of the source files. This parameter is used when **Filter Type** is set. **Input Path** is not used for filtering. If there are multiple filter conditions, use commas (,) to separate them. If the value is empty, the directories are not filtered.
	File Filter	Wildcard or regular expression for filtering the file names of the source files. This parameter is used when **Filter Type** is set. If there are multiple filter conditions, use commas (,) to separate them. The value cannot be left blank.

Source File Type	Parameter	Description
	Encoding Type	Source file encoding format, for example, UTF-8. This parameter can be set only in text file import.
	Suffix	File name extension added to a source file after the source file is imported. If this parameter is empty, no file name extension is added to the source file.
	Compression	Indicates whether to enable compressed transmission when SFTP is used to export data. **true** indicates that compression is enabled, and **false** indicates that compression is disabled.
hdfs-connector	Input Path	Input path of source files in HDFS
	Path Filter	Wildcard for filtering the directories in the input paths of the source files. **Input Path** is not used for filtering. If there are multiple filter conditions, use commas (,) to separate them. If the value is empty, the directories are not filtered. The regular expression filtering is not supported.
	File Filter	Wildcard for filtering the file names of the source files. If there are multiple filter conditions, use commas (,) to separate them. The value cannot be left blank. The regular expression filtering is not supported.
	Encoding Type	Source file encoding format, for example, UTF-8. This parameter can be set only in text file import.
	Suffix	File name extension added to a source file after the source file is imported. If this parameter is empty, no file name extension is added to the source file.
generic-jdbc-connector	Schema Name	Database schema name. This parameter exists in the **Table name** schema.
	Table Name	Database table name. This parameter exists in the **Table name** schema.

Source File Type	Parameter	Description
	SQL Statement	SQL statement for the Loader to query data to be imported in **Table SQL statement** mode. The SQL statement requires the query condition **WHERE ${CONDITIONS}**. Without this condition, the SQL statement cannot be run properly, for example, **select * from TABLE WHERE A>B and ${CONDITIONS}**. If **Table column names** is set, the column specified by **Table column names** will replace the column queried in the SQL statement. This parameter cannot be set when **Schema name** or **Table name** is set.
	Table Column Names	Table columns whose content is to be imported by Loader. Use commas (,) to separate multiple fields.
	Partition Column Name	Database table column based on which to-be-imported data is determined. This parameter is used for partitioning in a Map job. You are advised to configure the primary key field. **NOTE** • A partition column must have an index. If no index exists, do not specify a partition column. If a partition column without an index is specified, the database server disk I/O will be busy, the access of other services to the database will be affected, and the import will take a long period. • In multiple fields with indexes, select the field that has the most discrete value as the partition column. A partition column that is not discrete may result in load imbalance when multiple MapReduce jobs are imported. • The sorting rules of partition columns must be case-sensitive. Otherwise, data may be lost during data import. • You are not advised to select fields of the float or double type for the partition column. Otherwise, the records containing the minimum and maximum values of the partition column may fail to be imported due to precision issues.
	Nulls in Partition Column	Indicates whether to process records whose values are null in database table columns. If the value is **true**, the data whose value is null in the partition column is processed. If the value is **false**, the data whose value is null in the partition column is not processed.

Source File Type	Parameter	Description
	Whether to Specify a Partition Column	Indicates whether to specify a partition column.
oracle-connector	Table Name	Table name.
	Column Name	Column name.
	Query Condition	Query condition in an SQL statement
	Splitting Mode	Data splitting mode. The options are **ROWID** and **PARTITION**.
	Table Partition Name	Name of a table partition. Use commas (,) to separate the names of different partitions.
	Data Block Allocation Mode	Allocation method of data after being split.
	Read Size	Amount of data to be read each time.
mysql-fastpath-connector	Schema Name	Database schema name.
	Table Name	Database table name.
	Query Condition	Query condition of a specified table.
	Partition Column Name	Database table column based on which to-be-imported data is determined. This parameter is used for partitioning in a Map job. You are advised to configure the primary key field. **NOTE** • A partition column must have an index. If no index exists, do not specify a partition column. If a partition column without an index is specified, the database server disk I/O will be busy, the access of other services to the database will be affected, and the import will take a long period. • In multiple fields with indexes, select the field that has the most discrete value as the partition column. A partition column that is not discrete may result in load imbalance when multiple MapReduce jobs are imported. • You are not advised to select fields of the float or double type for the partition column. Otherwise, the records containing the minimum and maximum values of the partition column may fail to be imported due to precision issues.

Source File Type	Parameter	Description
	Nulls in Partition Column	Indicates whether to process records whose values are null in database table columns. If the value is **true**, the data whose value is null in the partition column is processed. If the value is **false**, the data whose value is null in the partition column is not processed.
	Whether to Specify a Partition Column	Indicates whether to specify a partition column.
oracle-partition-connector	Schema Name	Database schema name.
	Table Name	Partition table name.
	Query Condition	Query condition in an SQL statement.
	Table Column Names	Table columns whose content is to be imported by Loader. Use commas (,) to separate multiple fields.

Step 6 In the **3. Convert** area, set the conversion operations during data transmission.

Check whether source data values in the data operation job created by the Loader can be directly used without conversion, including upper and lower case conversion, cutting, merging, and separation.

- If yes, click **Next**.
- If no, perform **Step 6.1** to **Step 6.4**.

1. No created conversion step exists by default. Drag an example conversion step on the left to the edit box to create a new conversion step.
2. Conversion step types must be selected based on service requirements. A complete conversion process includes the following types:
   a. Input type. Only one conversion step can be added. This parameter is mandatory if the task involves HBase or relational databases.
   b. Conversion type, which is an intermediate conversion step. You can add one or more conversion types or do not add any conversion type.
   c. Output type. Only one output type can be added in the last conversion step. This parameter is mandatory if the task involves HBase or relational databases.

Table 20-4 Example list

Type	Description
Input Type	▪ **CSV File Input**: CSV file input step for configuring separators to generate multiple fields. ▪ **Fixed-Width File Input**: Text file input step for configuring the length of characters or bytes to be truncated to generate multiple fields. ▪ **Table Input**: relational data input step for configuring specified columns in the database as input fields. ▪ **HBase Input**: HBase table input step for configuring the column definition of an HBase table to a specified field. ▪ **HTML Input**: HTML web page data input step for obtaining the target data of the HTML web page file to the specified field. ▪ **Hive Input**: Hive table input step for defining columns in a Hive table to specified fields. ▪ **Spark Input**: Spark SQL table input step for defining columns in the SparkSQL table to specified fields. Only SparkSQL can access Hive data.

Type	Description
Conversion Type	▪ **Long Integer Time Conversion**: Configure the conversion between a long integer value and a date. ▪ **Null Value Conversion**: Configure a specified value to replace the null value. ▪ **Random Value Conversion**: Configure new value-added fields as random data fields. ▪ **Adding a Constant Field**: Add a constant to directly generate a constant field. ▪ **Concatenation and Conversion**: Concatenate fields, connect generated fields using connection characters, and convert new fields. ▪ **Separator Conversion**: Configure the generated fields to be separated by separators and convert new fields. ▪ **Modulo Conversion**: Configure the generated fields to be converted into new fields through modulo operation. ▪ **Cutting Character String**: Truncate a generated field based on a specified position to generate a new field. ▪ **EL Operation Conversion**: Calculate field values. Currently, the following operators are supported: md5sum, sha1sum, sha256sum, and sha512sum. ▪ **Character String Case Conversion:** Configure the generated fields to be converted to new fields through case conversion. ▪ **Reverse String Conversion**: Reverse the generated fields to generate new fields. ▪ **Character String Space Clearing Conversion**: Configure the generated fields to clear spaces and convert them to new fields. ▪ **Row Filtering Conversion**: Configure logical conditions to filter out rows that contain triggering conditions. ▪ **Update Fields**: Update the value of a specified field when certain conditions are met.

Type	Description
Output Type	▪ **File Output**: Configure generated fields to be connected by separators and exported to a file. ▪ **Table Output**: Configure the mapping between output fields and specified columns in the database. ▪ **HBase Output**: Configure the generated fields to the columns of the HBase table. ▪ **Hive Output**: Configure generated fields to a column of a Hive table. ▪ **Spark Output**: Configure generated fields to the columns of SparkSQL tables. Only SparkSQL can access Hive data.

The edit box allows you to perform the following tasks:

- Rename: Rename an example.
- Edit: Edit the step conversion by referring to **Step 6.3**.
- Delete: Delete an example.

  NOTE

  You can also use the shortcut key **Del** to delete the example.

3. Click **Edit** to edit the step conversion information and configure fields and data.

   For details about how to set parameters in the step conversion information, see **Loader Operator Help**.

   NOTE

   - When sftp-connector or ftp-connector is used to import data, the time type field in the original data must be set to a string during the data conversion so that the time can be accurate to millisecond for data import. The data that has more precise time than millisecond will not be imported.
   - When generic-jdbc-connector is used to import data, it is recommended that the data length of the CHAR or VARCHAR type field be set to -1 during data conversion so that all data can be imported. This prevents the data from being truncated when the actual data length is too long.
   - When generic-jdbc-connector is used to import data, the time type field in the original data must be set to a time type value during the data conversion so that the time can be accurate to second for data import. The data that has more precise time than second will not be imported.
   - When data is imported to a Hive partitioned table, Hive does not scan the newly imported data by default. You need to run the following HQL statement to repair the table so that the newly imported data can be queried:

     MSCK REPAIR TABLE *table_name*;

If the conversion step is incorrectly configured, the source data cannot be converted and become dirty data. The dirty data marking rules are as follows:

- In any input type step, all data becomes dirty data if the number of fields contained in the original data is less than that of configured fields, or the field values in the original data do not match the configured field type.
- In the **CSV File Input** step, **Validate input field** checks whether the input field matches the value type. If the input field and value type of a row do not match, the row is skipped and becomes dirty data.
- In the **Fixed Width File Input** step, **Fixed Length** specifies the field splitting length. If the length is greater than the length of the original field value, data splitting fails and the current row becomes dirty data.
- In the **HBase Input** step, if the HBase table name specified by **HBase Table Name** is incorrect, or no primary key column is configured for **Primary Key**, all data becomes dirty data.
- In any conversion step, rows whose conversion fails become dirty data. For example, in the **Split Conversion** step, if the number of generated fields is less than that of configured fields, or the original data cannot be converted to the String type, the current row becomes dirty data.
- In the **Filter Row Conversion** step, rows filtered by filter criteria become dirty data.
- In the **Modulo Conversion** step, if the original field value is **NULL**, the current row becomes dirty data.
- For jobs that import data to Hive/SparkSQL tables, you must configure the Hive conversion step.

4. Click **Next**.

Step 7 In the **4. Output Settings** area, set the destination location for saving data and click **Save** to save the job or click **Save and Run** to save and run the job.

Table 20-5 List of output configuration parameters

Storage Type	Parameter	Description
HDFS	File Type	The type of the file to be saved after data is imported to HDFS. Possible values are as follows: • **TEXT_FILE**: imports a text file and stores it as a text file. • **SEQUENCE_FILE**: imports a text file and stores it as a sequence file. • **BINARY_FILE**: imports files of any format by using binary streams but not to process the files. **NOTE** When the file import type to **TEXT_FILE** or **SEQUENCE_FILE**, Loader automatically selects a decompression method based on the file name extension to decompress a file.

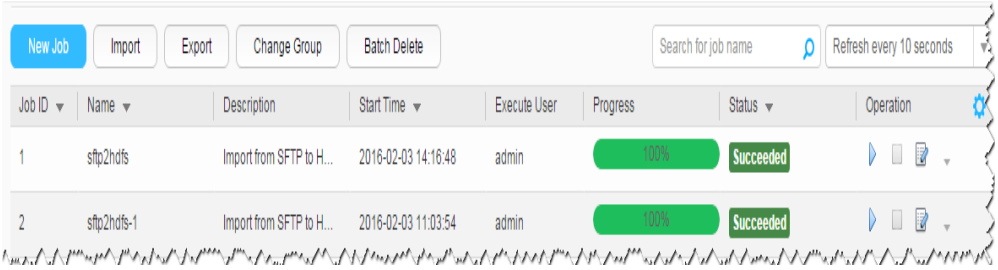
Storage Type	Parameter	Description
	Compression Format	Compression format of files imported to HDFS. Select a format from the drop-down list. If you select **NONE** or do not set this parameter, data is not compressed.
	Output Directory	Directory for storing data imported into HDFS.
	Operation	Action during data import. When all data is to be imported from the input path to the destination path, the data is stored in a temporary directory and then copied from the temporary directory to the destination path. After the data is imported successfully, the data is deleted from the temporary directory. One of the following actions can be taken when duplicate file names exist during data transfer: ● **OVERRIDE**: overrides the old file. ● **RENAME**: renames as new file. For a file without an extension, a string is added to the file name as the extension; for a file with an extension, a string is added to the extension. The string is unique. ● **APPEND**: adds the content of the new file to the end of the old file. This action only adds content regardless of whether the file can be used. For example, a text file can be used after this operation, while a compressed file cannot. ● **IGNORE**: reserves the old file and does not copy the new file. ● **ERROR**: stops the task and reports an error if duplicate file names exist. Transferred files are imported successfully, while files that have duplicate names and files that are not transferred fail to be imported.
	Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. This parameter cannot be set when **Extractor Size** is set. The value must be less than or equal to 3000.

Storage Type	Parameter	Description
	Extractor Size	Size of data processed by Maps that are started in a MapReduce task of a data configuration operation. The unit is MB. The value must be greater than or equal to 100. The recommended value is **1000**. This parameter cannot be set when **Extractors** is set. When a relational database connector is used, **Extractor Size** is unavailable. You need to set **Extractors**.
HBASE_BULKLOAD	HBase Instance	HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.
	Clear data before import	Indicates whether to clear data in the original table before importing data. The value **true** indicates that the clearing operation is performed, and the value **false** indicates that the clearing operation is not performed. If you do not set this parameter, the original table is not cleared by default.
	Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000.
	Extractor Size	HBase does not support this parameter. Please set **Extractors**.
HBASE_PUTLIST	HBase Instance	HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.
	Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000.
	Extractor Size	HBase does not support this parameter. Please set **Extractors**.
HIVE	Output Directory	Directory for storing data imported into Hive.

Storage Type	Parameter	Description
	Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. This parameter cannot be set when **Extractor Size** is set. The value must be less than or equal to 3000.
	Extractor Size	Size of data processed by Maps that are started in a MapReduce task of a data configuration operation. The unit is MB. The value must be greater than or equal to 100. The recommended value is **1000**. This parameter cannot be set when **Extractors** is set. When a relational database connector is used, **Extractor Size** is unavailable. You need to set **Extractors**.
SPARK	Output Directory	Only SparkSQL is supported to access Hive data. You can specify the directory for storing data imported to Hive.
	Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. This parameter cannot be set when **Extractor Size** is set. The value must be less than or equal to 3000.
	Extractor Size	Size of data processed by Maps that are started in a MapReduce task of a data configuration operation. The unit is MB. The value must be greater than or equal to 100. The recommended value is **1000**. This parameter cannot be set when **Extractors** is set. When a relational database connector is used, **Extractor Size** is unavailable. You need to set **Extractors**.

Step 8 On the Loader web UI, view, start, stop, copy, delete, edit, or view historical information about created jobs.

Figure 20-3 Viewing Loader jobs



----End

20.4.2 Using Loader to Import Data from an SFTP Server to HDFS or OBS

Scenario

Use Loader to import data from an SFTP server to HDFS or OBS.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the HDFS or OBS directories and data involved in job execution.
- You have obtained the username and password of the SFTP server as well as the read permission for the source files on the SFTP server. If file name extension needs to be added after a source file is imported, the user must have the write permission of the source file.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- When using Loader to import data from the SFTP server, the input paths and input path subdirectories of the SFTP server and the name of the files in these directories do not contain any of the special characters /"'\:;.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.

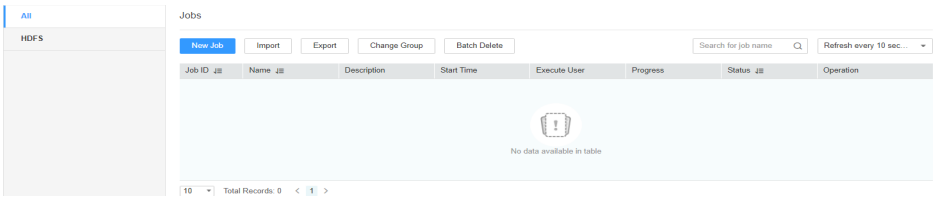
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-4 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-5 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Import**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **sftp-connector**, click **Add**, set connection parameters, and click **Test** to verify whether the connection is available. When "Test Success" is displayed, click **OK**. Loader allows multiple SFTP servers to be configured. Click **Add** to add the configuration information of multiple SFTP servers.

Table 20-6 Connection parameters

Parameter	Description	Example Value
Name	Name of the SFTP server connection	sftpName
SFTP Server IP Address	IP address of the SFTP server	10.16.0.1
SFTP Server Port	Port number of the SFTP server	22
SFTP Username	Username for accessing the SFTP server	root
SFTP Password	Password for accessing the SFTP server	xxxx

Parameter	Description	Example Value
SFTP Public Key	Public key of the SFTP server	OdDt/yn...etM

NOTE

When multiple SFTP servers are configured, the data in the specified directories of the SFTP servers is imported to the same directory in HDFS or OBS.

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set the data source information.

Table 20-7 Parameter description

Parameter	Description	Example Value
Input Path	Input path or name of the source file on an SFTP server. If multiple SFTP server IP addresses are configured for the connector, you can set this parameter to multiple input paths separated with semicolons (;). Ensure that the number of input paths is the same as that of SFTP servers configured for the connector. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/opt/tempfile;/opt
File Split Type	Indicates whether to split source files by file name or size. The files obtained after the splitting are used as the input files of each Map in the MapReduce task for data import. • **FILE**: indicates that the source file is split by file. That is, each Map processes one or multiple complete files, the same source file cannot be allocated to different Maps, and the source file directory structure is retained after data import. • **SIZE**: indicates that the source file is split by size. That is, each Map processes input files of a certain size, and a source file can be divided and processed by multiple Maps. After data is stored in the output directory, the number of saved files is the same as that of Maps. The file name format is **import_part**_*xxxx*, where *xxxx* is a unique random number generated by the system.	FILE

Parameter	Description	Example Value
Filter Type	File filter condition. This parameter is used when **Path Filter** or **File Filter** is set. • **WILDCARD**: indicates using a wildcard. • **REGEX**: indicates using a regular expression. • If the parameter is not set, a wildcard is used by default.	WILDCARD
Path Filter	Wildcard or regular expression for filtering the directories in the input path of the source files. This parameter is used when **Filter Type** is set. **Input Path** is not used for filtering. Use semicolons (;) to separate the path filters on multiple servers and use commas (,) to separate the filter conditions of each server. If this parameter is left empty, directories are not filtered. • **?** matches a single character. • ***** indicates multiple characters. • Adding ^ before the condition indicates negated filtering, that is, file filtering. For example, when **Filter type** is set to **WILDCARD**, set the parameter to *****; when **Filter type** is set to **REGEX**, set the parameter to **\\.***.	1*,2*;1*
File Filter	Wildcard or regular expression for filtering the file names of the source files. This parameter is used when **Filter Type** is set. Use semicolons (;) to separate the path filters on multiple servers and use commas (,) to separate the filter conditions of each server. This parameter cannot be left blank. • **?** matches a single character. • ***** indicates multiple characters. • Adding ^ before the condition indicates negated filtering, that is, file filtering. For example, when **Filter type** is set to **WILDCARD**, set the parameter to *****; when **Filter type** is set to **REGEX**, set the parameter to **\\.***.	*.txt,*.csv; *.txt
Encoding Type	Source file encoding format, for example, UTF-8 and GBK. This parameter can be set only in text file import.	UTF-8

Parameter	Description	Example Value
Suffix	File name extension added to a source file after the source file is imported. If this parameter is empty, no file name extension is added to the source file. This parameter is valid only when the data source is a file system. You are advised to set this parameter in incremental data import. For example, if the parameter is set to **.txt** and the source file is **test-loader.csv**, the source file name is **test-loader.csv.txt** after export.	.log
Compression	Indicates whether to enable compressed transmission when SFTP is used to export data. • The value **true** indicates that compression is enabled. • The value **false** indicates that compression is disabled.	true

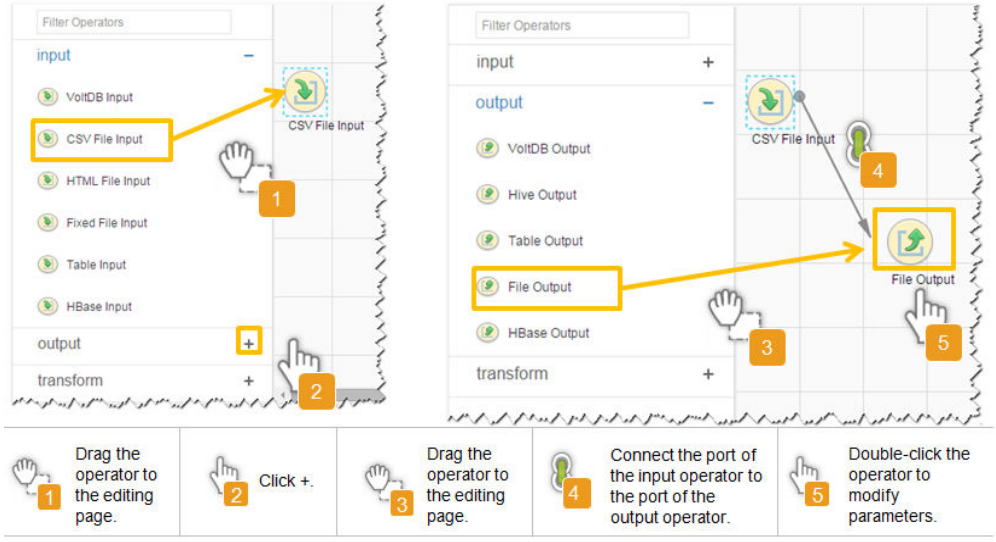
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-8**.

Table 20-8 Input and output parameters of the operator

Input Type	Output Type
CSV File Input	File Output
HTML Input	File Output
Fixed File Input	File Output

Figure 20-6 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set **Storage type** to **HDFS**.

Table 20-9 Parameter description

Parameter	Description	Example Value
File Type	Type of a file after the file is imported. The options are as follows: • **TEXT_FILE**: imports a text file and stores it as a text file. • **SEQUENCE_FILE**: imports a text file and stores it as a sequence file. • **BINARY_FILE**: imports files of any format by using binary streams.	TEXT_FILE
Compression Format	Compression format of files imported to HDFS or OBS. Select a format from the drop-down list. If you select **NONE** or do not set this parameter, data is not compressed.	NONE
Output Directory	Directory for storing data imported into HDFS or OBS. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/user/test
Delete Files Before Execution (This parameter is available for MRS 3.5.0 or later.)	Whether to delete the files generated based on the file name identifier before the job is executed so that data can be reloaded. • **false** (default): The files are not deleted before the job is executed. • **true**: Specified files are deleted before the job is executed. If the **Operation** is set to **OVERRIDE**, the files overwrite the old files. Only the following connectors are supported: – generic-jdbc-connector – oracle-connector – mysql-fastpath-connector	false

Parameter	Description	Example Value
File Name Identifier (This parameter is available for MRS 3.5.0 or later.)	File name identifier, which is used to generate the name of the loaded file. ● File name format: **import_part**_*xxx*_**000000000** – *xxx*: file name – Value range: [0-9], [a-z], [A-Z], hyphen (-), underscore (_), and dot (.) – Value length: [1-255] ● This parameter is mandatory when **Delete Files Before Execution** is **true**.	12
Operation	Action during data import. When all data is to be imported from the input path to the destination path, the data is stored in a temporary directory and then copied from the temporary directory to the destination path. After the data is imported successfully, the data is deleted from the temporary directory. One of the following actions can be taken when duplicate file names exist during data transfer: ● **OVERRIDE**: overrides the old file. ● **RENAME**: renames as new file. For a file without an extension, a string is added to the file name as the extension; for a file with an extension, a string is added to the extension. The string is unique. ● **APPEND**: adds the content of the new file to the end of the old file. This action only adds content regardless of whether the file can be used. For example, a text file can be used after this operation, while a compressed file cannot. ● **IGNORE**: reserves the old file and does not copy the new file. ● **ERROR**: stops the task and reports an error if duplicate file names exist. Transferred files are imported successfully, while files that have duplicate names and files that are not transferred fail to be imported.	OVERRIDE

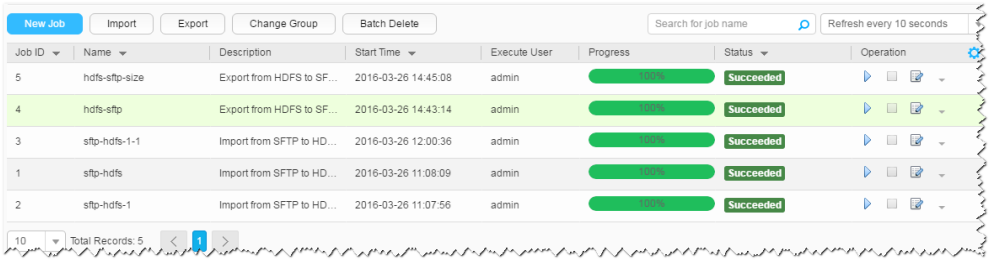
Parameter	Description	Example Value
Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. This parameter cannot be set when **Extractor Size** is set. The value must be less than or equal to 3000. You are advised to set the parameter to the number of CPU cores on the SFTP server. **NOTE** To improve the data import speed, ensure that the following conditions are met: • Each Map connection is equivalent to a client connection. Therefore, you must ensure that the maximum number of connections of the SFTP server is greater than the number of Maps. • Ensure that the disk I/O or network bandwidth on the SFTP server does not reach the upper limit.	20
Extractor Size	Size of data processed by Maps that are started in a MapReduce task of a data configuration operation. The unit is MB. The value must be greater than or equal to 100. The recommended value is 1000. This parameter cannot be set when **Extractors** is set.	1000

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the Loader web UI. When **Status** is **Succeeded**, the job is complete.

Figure 20-7 Viewing job details



----End

20.4.3 Using Loader to Import Data from an SFTP Server to HBase

Scenario

Use Loader to import data from an SFTP server to HBase.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the HBase tables or phoenix tables that are used during job execution.
- You have obtained the username and password of the SFTP server as well as the read permission for the source files on the SFTP server. If file name extension needs to be added after a source file is imported, the user must have the write permission of the source file.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- When using Loader to import data from the SFTP server, the input paths and input path subdirectories of the SFTP server and the name of the files in these directories do not contain any of the special characters /"':;.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.

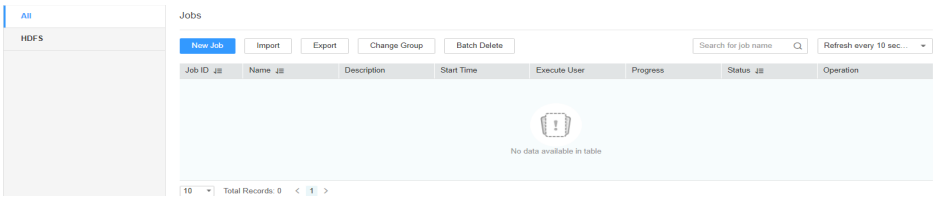
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-8 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-9 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Import**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **sftp-connector**, click **Add**, set connection parameters, and click **Test** to verify whether the connection is available. When "Test Success" is displayed, click **OK**. Loader allows multiple SFTP servers to be configured. Click **Add** to add the configuration information of multiple SFTP servers.

Table 20-10 Connection parameters

Parameter	Description	Example Value
Name	Name of the SFTP server connection	sftpName
SFTP Server IP Address	IP address of the SFTP server	10.16.0.1
SFTP Server Port	Port number of the SFTP server	22
SFTP Username	Username for accessing the SFTP server	root
SFTP Password	Password for accessing the SFTP server	xxxx
SFTP Public Key	Public key of the SFTP server	OdDt/yn...etM

NOTE

When multiple SFTP servers are configured, the data in the specified directories of the servers is imported to HBase.

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set the data source information.

Table 20-11 Parameter description

Parameter	Description	Example Value
Input Path	Input path or name of the source file on an SFTP server. If multiple SFTP server IP addresses are configured for the connector, you can set this parameter to multiple input paths separated with semicolons (;). Ensure that the number of input paths is the same as that of SFTP servers configured for the connector. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/opt/tempfile;/opt
File Split Type	Indicates whether to split source files by file name or size. The files obtained after the splitting are used as the input files of each Map in the MapReduce task for data import. • **FILE**: indicates that the source file is split by file. That is, each Map processes one or multiple complete files, the same source file cannot be allocated to different Maps, and the source file directory structure is retained after data import. • **SIZE**: indicates that the source file is split by size. That is, each Map processes input files of a certain size, and a source file can be divided and processed by multiple Maps. After data is stored in the output directory, the number of saved files is the same as that of Maps. The file name format is **import_part**_*xxxx*, where *xxxx* is a unique random number generated by the system.	FILE
Filter Type	File filter condition. This parameter is used when **Path Filter** or **File Filter** is set. • **WILDCARD**: indicates using a wildcard. • **REGEX**: indicates using a regular expression. • If the parameter is not set, a wildcard is used by default.	WILDCARD

Parameter	Description	Example Value
Path Filter	Wildcard or regular expression for filtering the directories in the input path of the source files. This parameter is used when **Filter Type** is set. **Input Path** is not used for filtering. Use semicolons (;) to separate the path filters on multiple servers and use commas (,) to separate the filter conditions of each server. If this parameter is left empty, directories are not filtered. ● **?** matches a single character. ● ***** indicates multiple characters. ● Adding **^** before the condition indicates negated filtering, that is, file filtering. For example, when **Filter type** is set to **WILDCARD**, set the parameter to *****; when **Filter type** is set to **REGEX**, set the parameter to **\\.***.	1*,2*;1*
File Filter	Wildcard or regular expression for filtering the file names of the source files. This parameter is used when **Filter Type** is set. Use semicolons (;) to separate the path filters on multiple servers and use commas (,) to separate the filter conditions of each server. This parameter cannot be left blank. ● **?** matches a single character. ● ***** indicates multiple characters. ● Adding **^** before the condition indicates negated filtering, that is, file filtering. For example, when **Filter type** is set to **WILDCARD**, set the parameter to *****; when **Filter type** is set to **REGEX**, set the parameter to **\\.***.	*.txt,*.csv;*.txt
Encoding Type	Source file encoding format, for example, UTF-8 and GBK. This parameter can be set only in text file import.	UTF-8

Parameter	Description	Example Value
Suffix	File name extension added to a source file after the source file is imported. If this parameter is empty, no file name extension is added to the source file. This parameter is valid only when the data source is a file system. You are advised to set this parameter in incremental data import. For example, if the parameter is set to **.txt** and the source file is **test-loader.csv**, the source file name is **test-loader.csv.txt** after export.	.log
Compression	Indicates whether to enable compressed transmission when SFTP is used to export data. • The value **true** indicates that compression is enabled. • The value **false** indicates that compression is disabled.	true

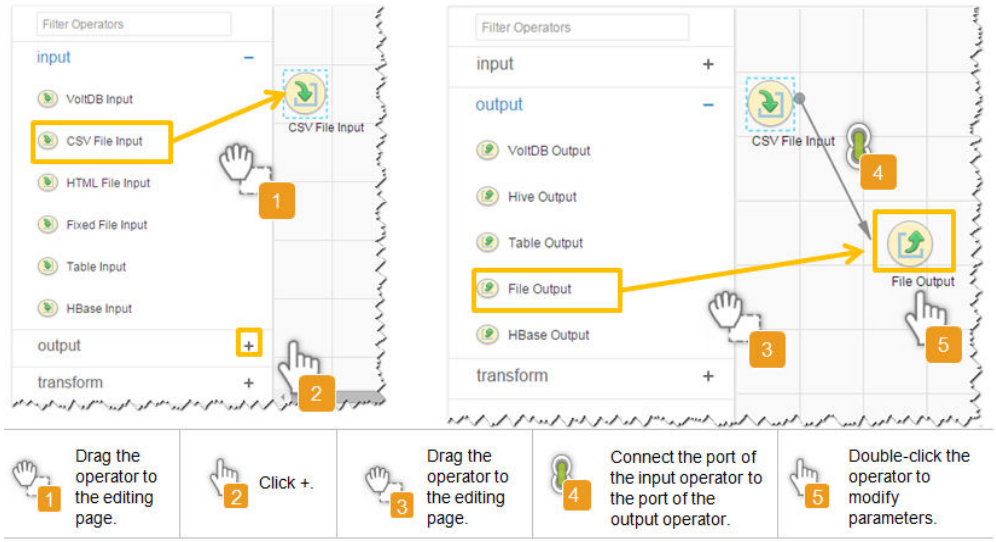
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-12**.

Table 20-12 Input and output parameters of the operator

Input Type	Output Type
CSV File Input	HBase Output
HTML Input	HBase Output
Fixed File Input	HBase Output

Figure 20-10 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set **Storage type** to **HBASE_BULKLOAD** or **HBASE_PUTLIST** based on the actual situation.

Table 20-13 Parameter description

Storage Type	Applicable Scenario	Parameter	Description	Example Value
HBASE_BULKLOAD	Large data volume	HBase Instance	HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.	HBase

Storage Type	Applicable Scenario	Parameter	Description	Example Value
		Clear data before import	Indicates whether to clear data in the original table before importing data. The value **true** indicates that the data is cleared, and the value **false** indicates that the data is not cleared. If you do not set this parameter, the original table is not cleared by default.	true
		Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000. You are advised to set the parameter to the maximum number of connections on the SFTP server.	20
		Extractor Size	HBase does not support this parameter. Please set **Extractors**.	-

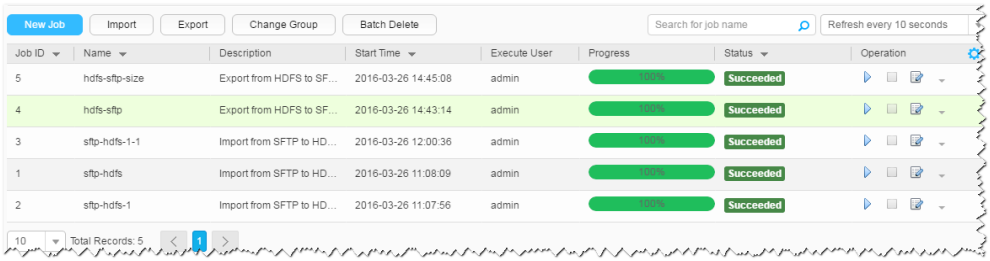
Storage Type	Applicable Scenario	Parameter	Description	Example Value
HBASE_PUTLIST	Small data volume	HBase Instance	HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.	HBase
		Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000.	20
		Extractor Size	HBase does not support this parameter. Please set **Extractors**.	-

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the Loader web UI. When **Status** is **Succeeded**, the job is complete.

Figure 20-11 Viewing job details



----End

20.4.4 Using Loader to Import Data from an SFTP Server to Hive

Scenario

Use Loader to import data from an SFTP server to Hive.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the Hive table specified in the job.
- You have obtained the username and password of the SFTP server as well as the read permission for the source files on the SFTP server. If file name extension needs to be added after a source file is imported, the user must have the write permission of the source file.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- When using Loader to import data from the SFTP server, the input paths and input path subdirectories of the SFTP server and the name of the files in these directories do not contain any of the special characters /"':;.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.

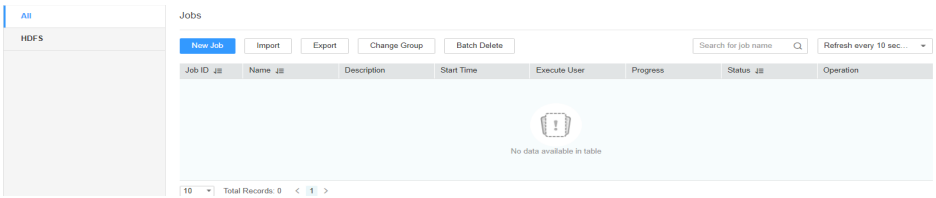
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-12 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-13 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Import**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **sftp-connector**, click **Add**, set connection parameters, and click **Test** to verify whether the connection is available. When "Test Success" is displayed, click **OK**. Loader allows multiple SFTP servers to be configured. Click **Add** to add the configuration information of multiple SFTP servers.

Table 20-14 Connection parameters

Parameter	Description	Example Value
Name	Name of the SFTP server connection	sftpName
SFTP Server IP Address	IP address of the SFTP server	10.16.0.1
SFTP Server Port	Port number of the SFTP server	22
SFTP Username	Username for accessing the SFTP server	root
SFTP Password	Password for accessing the SFTP server	xxxx

Parameter	Description	Example Value
SFTP Public Key	Public key of the SFTP server	OdDt/yn...etM

NOTE

When multiple SFTP servers are configured, the data in the specified directories of the servers is imported to Hive.

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set the data source information.

Table 20-15 Parameter description

Parameter	Description	Example Value
Input Path	Input path or name of the source file on an SFTP server. If multiple SFTP server IP addresses are configured for the connector, you can set this parameter to multiple input paths separated with semicolons (;). Ensure that the number of input paths is the same as that of SFTP servers configured for the connector. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/opt/tempfile;/opt
File Split Type	Indicates whether to split source files by file name or size. The files obtained after the splitting are used as the input files of each Map in the MapReduce task for data import. • **FILE**: indicates that the source file is split by file. That is, each Map processes one or multiple complete files, the same source file cannot be allocated to different Maps, and the source file directory structure is retained after data import. • **SIZE**: indicates that the source file is split by size. That is, each Map processes input files of a certain size, and a source file can be divided and processed by multiple Maps. After data is stored in the output directory, the number of saved files is the same as that of Maps. The file name format is **import_part**_*xxxx*, where *xxxx* is a unique random number generated by the system.	FILE

Parameter	Description	Example Value
Filter Type	File filter condition. This parameter is used when **Path Filter** or **File Filter** is set. ● **WILDCARD**: indicates using a wildcard. ● **REGEX**: indicates using a regular expression. ● If the parameter is not set, a wildcard is used by default.	WILDCARD
Path Filter	Wildcard or regular expression for filtering the directories in the input path of the source files. This parameter is used when **Filter Type** is set. **Input Path** is not used for filtering. Use semicolons (;) to separate the path filters on multiple servers and use commas (,) to separate the filter conditions of each server. If this parameter is left empty, directories are not filtered. ● **?** matches a single character. ● ***** indicates multiple characters. ● Adding ^ before the condition indicates negated filtering, that is, file filtering. For example, when **Filter type** is set to **WILDCARD**, set the parameter to *****; when **Filter type** is set to **REGEX**, set the parameter to **\\.***.	1*,2*;1*
File Filter	Wildcard or regular expression for filtering the file names of the source files. This parameter is used when **Filter Type** is set. Use semicolons (;) to separate the path filters on multiple servers and use commas (,) to separate the filter conditions of each server. This parameter cannot be left blank. ● **?** matches a single character. ● ***** indicates multiple characters. ● Adding ^ before the condition indicates negated filtering, that is, file filtering. For example, when **Filter type** is set to **WILDCARD**, set the parameter to *****; when **Filter type** is set to **REGEX**, set the parameter to **\\.***.	*.txt,*.csv;*.txt
Encoding Type	Source file encoding format, for example, UTF-8 and GBK. This parameter can be set only in text file import.	UTF-8

Parameter	Description	Example Value
Suffix	File name extension added to a source file after the source file is imported. If this parameter is empty, no file name extension is added to the source file. This parameter is valid only when the data source is a file system. You are advised to set this parameter in incremental data import. For example, if the parameter is set to **.txt** and the source file is **test-loader.csv**, the source file name is **test-loader.csv.txt** after export.	.log
Compression	Indicates whether to enable compressed transmission when SFTP is used to export data. • The value **true** indicates that compression is enabled. • The value **false** indicates that compression is disabled.	true

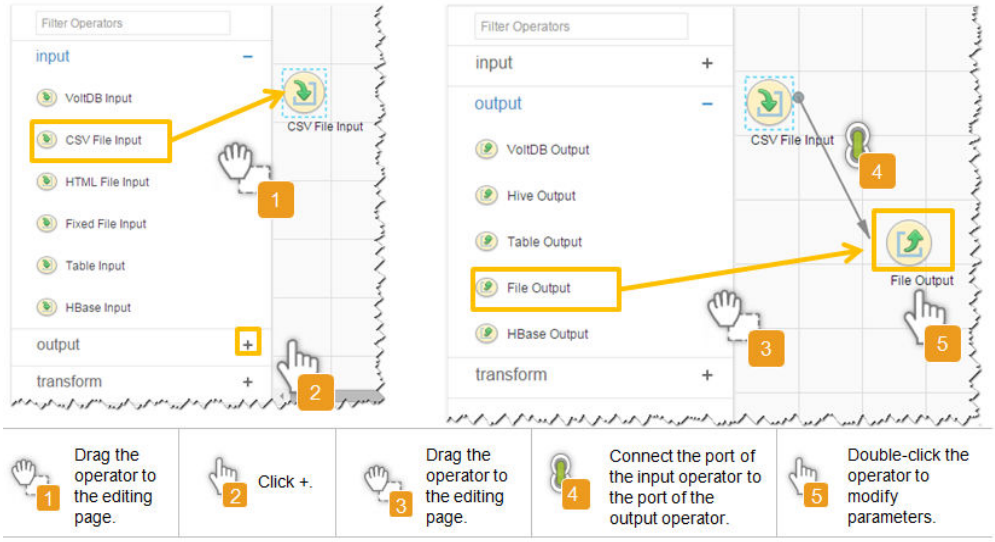
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-16**.

Table 20-16 Input and output parameters of the operator

Input Type	Output Type
CSV File Input	Hive Output
HTML Input	Hive Output
Fixed File Input	Hive Output

Figure 20-14 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set **Storage type** to **HIVE**.

Table 20-17 Parameter description

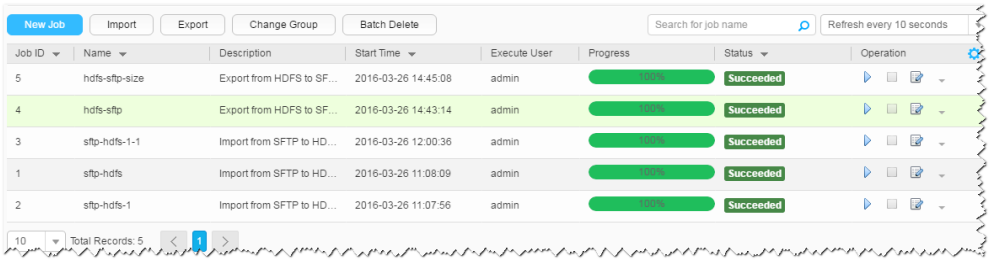
Parameter	Description	Example Value
Output Directory	Directory for storing data imported into Hive. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/opt/tempfile
Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000. You are advised to set the parameter to the maximum number of connections on the SFTP server.	20
Extractor Size	Hive does not support this parameter. Please set **Extractors**.	-

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the Loader web UI. When **Status** is **Succeeded**, the job is complete.

Figure 20-15 Viewing job details



----End

20.4.5 Using Loader to Import Data from an FTP Server to HBase

Scenario

Use Loader to import data from an FTP server to HBase.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have obtained the username and password of the FTP server and the user has the read permission of the source files on the FTP server. If file name extension needs to be added after a source file is imported, the user must have the write permission of the source file.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- When using Loader to import data from the FTP server, the input paths and input path subdirectories of the FTP server and the name of the files in these directories do not contain any of the special characters /"'\:;.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.

Procedure

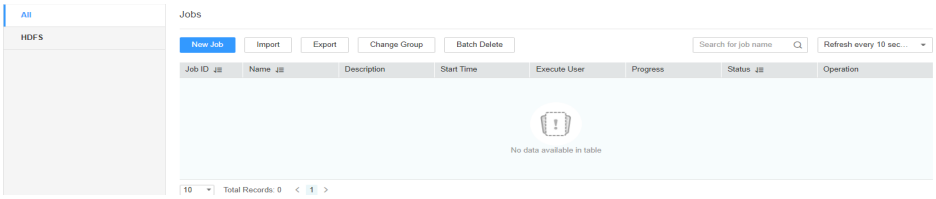
Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.

2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-16 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-17 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Import**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **ftp-connector**, click **Add**, set connection parameters, and click **Test** to verify whether the connection is available. When "Test Success" is displayed, click **OK**. Loader allows multiple FTP servers to be configured. Click **Add** to add the configuration information of multiple FTP servers.

Table 20-18 Connection parameters

Parameter	Description	Example Value
FTP Server IP Address	IP address of the FTP server	ftpName
FTP Server Port	Port number of the FTP server	22

Parameter	Description	Example Value
FTP Username	Username for accessing the FTP server	root
FTP Password	Password for accessing the FTP server	xxxx
FTP Mode	FTP access mode. Possible values are **ACTIVE** and **PASSIVE**. If this parameter is not set, FTP access is in passive mode by default.	PASSIVE
FTP Protocol	FTP protocol. • **FTP**: indicates the FTP protocol. • **SSL_EXPLICIT**: indicates the explicit SSL protocol. • **SSL_IMPLICIT**: indicates the implicit SSL protocol. • **TLS_EXPLICIT**: indicates the explicit TLS protocol. • **TLS_IMPLICIT**: indicates the implicit TLS protocol. If this parameter is not set, the FTP protocol is used by default.	FTP
File Name Encoding Type	File name and file path encoding format supported by the FTP server. If this parameter is not set, the default format UTF-8 is used.	UTF-8

NOTE

When multiple FTP servers are configured, the data in the specified directories of the servers is imported to HBase.

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set the data source information.

Table 20-19 Parameter description

Parameter	Description	Example Value
Input Path	Input path or name of the source file on an FTP server. If multiple FTP server IP addresses are configured for the connector, you can set this parameter to multiple input paths separated with semicolons (**;**). Ensure that the number of input paths is the same as that of FTP servers configured for the connector. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/opt/tempfile;/opt
File Split Type	Indicates whether to split source files by file name or size. The files obtained after the splitting are used as the input files of each Map in the MapReduce task for data import. • **FILE**: indicates that the source file is split by file. That is, each Map processes one or multiple complete files, the same source file cannot be allocated to different Maps, and the source file directory structure is retained after data import. • **SIZE**: indicates that the source file is split by size. That is, each Map processes input files of a certain size, and a source file can be divided and processed by multiple Maps. After data is stored in the output directory, the number of saved files is the same as that of Maps. The file name format is **import_part**_*xxxx*, where *xxxx* is a unique random number generated by the system.	FILE
Filter Type	File filter condition. This parameter is used when **Path Filter** or **File Filter** is set. • **WILDCARD**: indicates using a wildcard. • **REGEX**: indicates using a regular expression. • If the parameter is not set, a wildcard is used by default.	WILDCARD

Parameter	Description	Example Value
Path Filter	Wildcard or regular expression for filtering the directories in the input path of the source files. This parameter is used when **Filter Type** is set. **Input Path** is not used for filtering. Use semicolons (;) to separate the path filters on multiple servers and use commas (,) to separate the filter conditions of each server. If this parameter is left empty, directories are not filtered. ● **?** matches a single character. ● ***** indicates multiple characters. ● Adding **^** before the condition indicates negated filtering, that is, file filtering. For example, when **Filter type** is set to **WILDCARD**, set the parameter to *****; when **Filter type** is set to **REGEX**, set the parameter to **\\.***.	1*,2*;1*
File Filter	Wildcard or regular expression for filtering the file names of the source files. This parameter is used when **Filter Type** is set. Use semicolons (;) to separate the path filters on multiple servers and use commas (,) to separate the filter conditions of each server. This parameter cannot be left blank. ● **?** matches a single character. ● ***** indicates multiple characters. ● Adding **^** before the condition indicates negated filtering, that is, file filtering. For example, when **Filter type** is set to **WILDCARD**, set the parameter to *****; when **Filter type** is set to **REGEX**, set the parameter to **\\.***.	*.txt,*.csv;*.txt
Encoding Type	Source file encoding format, for example, UTF-8 and GBK. This parameter can be set only in text file import.	UTF-8

Parameter	Description	Example Value
Suffix	File name extension added to a source file after the source file is imported. If this parameter is empty, no file name extension is added to the source file. This parameter is valid only when the data source is a file system. You are advised to set this parameter in incremental data import. For example, if the parameter is set to **.txt** and the source file is **test-loader.csv**, the source file name is **test-loader.csv.txt** after export.	.log
Compression	Indicates whether to enable compressed transmission when FTP is used to export data. • The value **true** indicates that compression is enabled. • The value **false** indicates that compression is disabled.	true

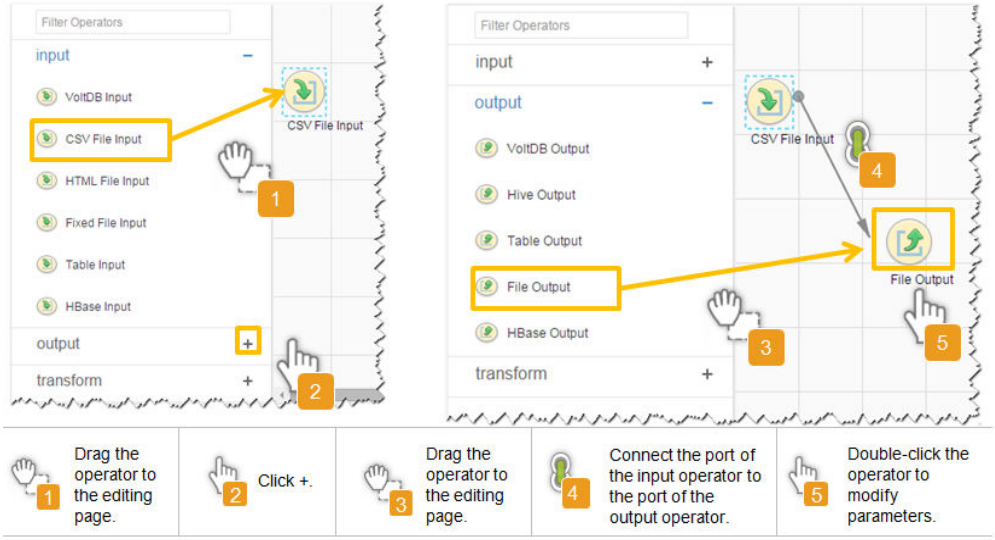
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-20**.

Table 20-20 Input and output parameters of the operator

Input Type	Output Type
CSV File Input	HBase Output
HTML Input	HBase Output
Fixed File Input	HBase Output

Figure 20-18 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set **Storage type** to **HBASE_BULKLOAD** or **HBASE_PUTLIST** based on the actual situation.

Table 20-21 Parameter description

Storage Type	Applicable Scenario	Parameter	Description	Example Value
HBASE_BULKLOAD	Large data volume	HBase Instance	HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.	HBase
		Clear data before import	Indicates whether to clear data in the original table before importing data. **True** indicates clearing data and **False** indicates not to clear data. If you do not set this parameter, the original table is not cleared by default.	true

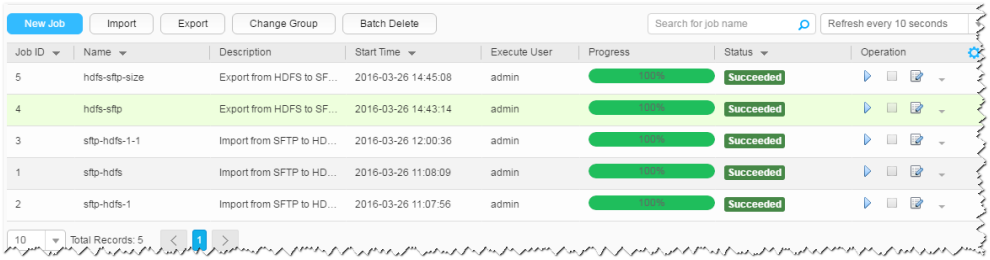
Storage Type	Applicable Scenario	Parameter	Description	Example Value
		Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000. You are advised to set the parameter to the maximum number of connections on the FTP server.	20
		Extractor Size	HBase does not support this parameter. Please set **Extractors**.	-
HBASE_PUTLIST	Small data volume	HBase Instance	HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.	HBase
		Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000.	20
		Extractor Size	HBase does not support this parameter. Please set **Extractors**.	-

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the Loader web UI. When **Status** is **Succeeded**, the job is complete.

Figure 20-19 Viewing job details



----End

20.4.6 Using Loader to Import Data from a Relational Database to HDFS or OBS

Scenario

Use Loader to import data from a relational database to HDFS or OBS.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the HDFS or OBS directories and data involved in job execution.
- You have obtained the username and password of the relational database.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.
- Before the operation, perform the following steps:
  a. Obtain the JAR package of the relational database driver and save it to the following directory on the active and standby Loader nodes: **${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib**.
  b. Run the following command on the active and standby nodes as user **root** to modify the permission:

     cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

     chown omm:wheel *JAR package name*

     chmod 600 *JAR package name*
  c. Log in to FusionInsight Manager. Choose **Cluster** > **Services** > **Loader** > **More** > **Restart Service**. Enter the password of the administrator to restart the Loader service.

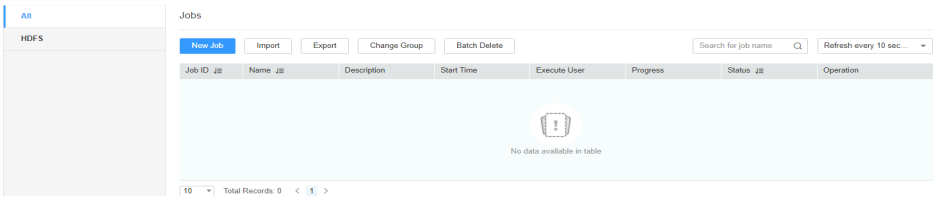
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-20 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-21 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Import**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **generic-jdbc-connector** or a dedicated database connector (oracle-connector, oracle-partition-connector, or mysql-fastpath-connector), set connection parameters, and click **Test** to verify whether the connection is available. When "Test Success" is displayed, click **OK**.

NOTE

- For connection to relational databases, general database connectors (generic-jdbc-connector) or dedicated database connectors (oracle-connector, oracle-partition-connector, and mysql-fastpath-connector) are available. However, compared with general database connectors, dedicated database connectors perform better in data import and export because they are optimized for specific database types.
- When mysql-fastpath-connector is used, the **mysqldump** and **mysqlimport** commands of MySQL must be available on NodeManager nodes, and the MySQL client version to which the two commands belong must be compatible with the MySQL server version. If the two commands are unavailable or the versions are incompatible, install the MySQL client applications and tools following the instructions at **http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**.

Table 20-22 generic-jdbc-connector connection parameters

Parameter	Description	Example Value
Name	Name of a relational database connection	dbName
JDBC Driver Class	Name of a Java database connectivity (JDBC) driver class	oracle.jdbc.driver.OracleDriver
JDBC Connection String	JDBC connection string	jdbc:oracle:thin:@//10.16.0.1:1521/oradb
Username	Username for connecting to the database	omm
Password	Password for connecting to the database	xxxx
JDBC Connection Properties	JDBC connection attribute. Click **Add** to manually add the attribute. • Name: connection attribute name • Value: connection attribute value	• Name: **socketTimeout** • Value: **20**

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set the data source information.

Table 20-23 Parameter description

Parameter	Description	Example Value
Schema Name	Database schema name. This parameter exists in the **Table name** schema.	public
Table Name	Database table name. This parameter exists in the **Table name** schema.	test

Parameter	Description	Example Value
SQL Statement	SQL statement for Loader to query data to be imported in **Table SQL statement** mode. The SQL statement requires the query condition **WHERE ${CONDITIONS}**. Without this condition, the SQL statement cannot be run properly. An example SQL statement is as follows: **select * from TABLE WHERE A>B and ${CONDITIONS}**. If **Table column names** is set, the column specified by **Table column names** will replace the column queried in the SQL statement. This parameter cannot be set when **Schema name** or **Table name** is set. **NOTE** You can use macros to define SQL Where statements. For details, see **Using Macro Definitions in Configuration Items** .	select * from TABLE WHERE A>B and ${CONDITIONS}
Table Column Names	Table columns whose content is to be imported by Loader. Use commas (**,**) to separate multiple fields. If the parameter is not set, all the columns are imported and the **Select *** order is used as the column location.	id,name
Partition Column Name	Database table column based on which to-be-imported data is determined. This parameter is used for partitioning in a Map job. You are advised to configure the primary key field. **NOTE** • A partition column must have an index. If no index exists, do not specify a partition column. If a partition column without an index is specified, the database server disk I/O will be busy, the access of other services to the database will be affected, and the import will take a long period. • In multiple fields with indexes, select the field that has the most discrete value as the partition column. A partition column that is not discrete may result in load imbalance when multiple MapReduce jobs are imported. • The sorting rules of partition columns must be case-sensitive. Otherwise, data may be lost during data import. • You are not advised to select fields of the float or double type for the partition column. Otherwise, the records containing the minimum and maximum values of the partition column may fail to be imported due to precision issues.	id

Parameter	Description	Example Value
Nulls in Partition Column	Indicates whether to process records whose values are null in database table columns. • **true**: Records whose values are null are processed. • **false**: Records whose values are not null are processed.	true
Whether to Specify a Partition Column	Indicates whether to specify a partition column.	true

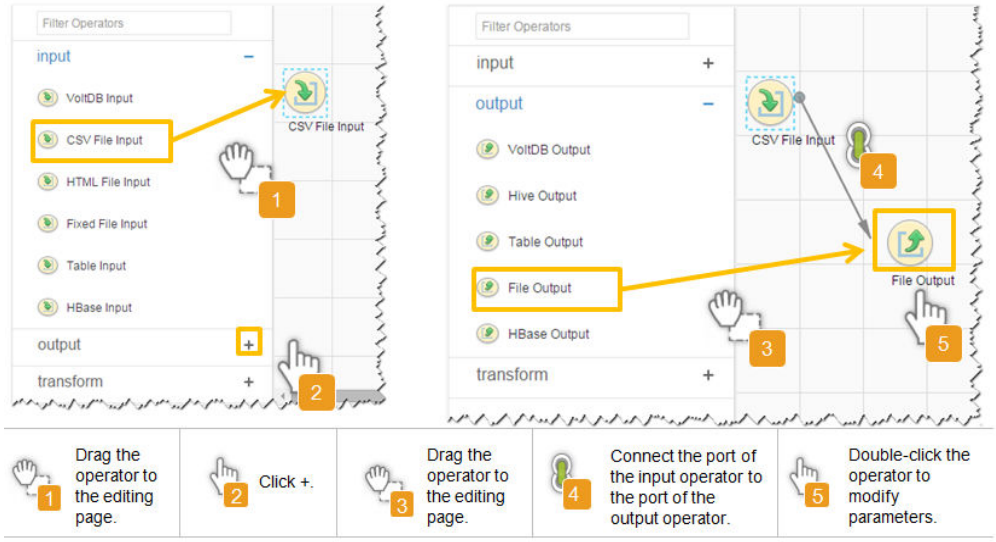
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-24**.

Table 20-24 Input and output parameters of the operator

Input Type	Output Type
Table input	File output

Figure 20-22 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set **Storage type** to **HDFS**.

Table 20-25 Parameter description

Parameter	Description	Example Value
File Type	Type of a file after the file is imported. The options are as follows: • **TEXT_FILE**: imports a text file and stores it as a text file. • **SEQUENCE_FILE**: imports a text file and stores it as a sequence file. • **BINARY_FILE**: imports files of any format by using binary streams.	TEXT_FILE
Compression Format	Compression format of files imported to HDFS or OBS. Select a format from the drop-down list. If you select **NONE** or leave this parameter blank, data is not compressed.	NONE
Output Directory	Directory for storing data imported into HDFS or OBS. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/user/test
Delete Files Before Execution (This parameter is available for MRS 3.5.0 or later.)	Whether to delete the files generated based on the file name identifier before the job is executed so that data can be reloaded. • **false** (default): The files are not deleted before the job is executed. • **true**: Specified files are deleted before the job is executed. If the **Operation** is set to **OVERRIDE**, the files overwrite the old files. Only the following connectors are supported: – generic-jdbc-connector – oracle-connector – mysql-fastpath-connector	false
File Name Identifier (This parameter is available for MRS 3.5.0 or later.)	File name identifier, which is used to generate the name of the loaded file. • File name format: **import_part_***xxx*_**000000000** – *xxx*: file name – Value range: [0-9], [a-z], [A-Z], hyphen (-), underscore (_), and dot (.) – Value length: [1-255] • This parameter is mandatory when **Delete Files Before Execution** is **true**.	12

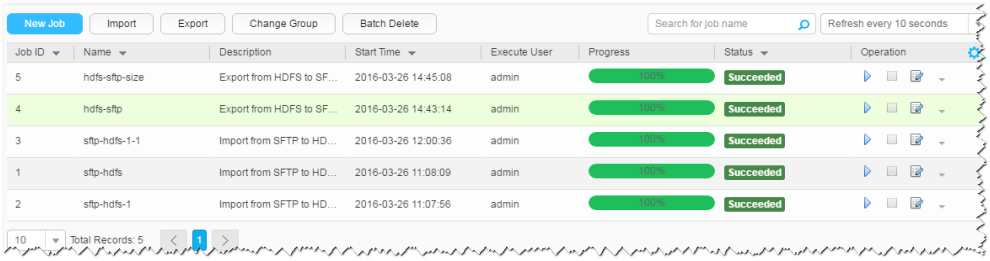
Parameter	Description	Example Value
Operation	Action during data import. When all data is to be imported from the input path to the destination path, the data is stored in a temporary directory and then copied from the temporary directory to the destination path. After the data is imported successfully, the data is deleted from the temporary directory. One of the following actions can be taken when duplicate file names exist during data transfer: ● **OVERRIDE**: overrides the old file. ● **RENAME**: renames as new file. For a file without an extension, a string is added to the file name as the extension; for a file with an extension, a string is added to the extension. The string is unique. ● **APPEND**: adds the content of the new file to the end of the old file. This action only adds content regardless of whether the file can be used. For example, a text file can be used after this operation, while a compressed file cannot. ● **IGNORE**: reserves the old file and does not copy the new file. ● **ERROR**: stops the task and reports an error if duplicate file names exist. Transferred files are imported successfully, while files that have duplicate names and files that are not transferred fail to be imported.	OVERRIDE
Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. This parameter cannot be set when **Extractor Size** is set. The value must be less than or equal to 3000.	-
Extractor Size	Size of data processed by Maps that are started in a MapReduce task of a data configuration operation. The unit is MB. The value must be greater than or equal to 100. The recommended value is 1000. This parameter cannot be set when **Extractors** is set. When a relational database connector is used, **Extractor Size** is unavailable. You need to set **Extractors**.	1000

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the Loader web UI. When **Status** is **Succeeded**, the job is complete.

Figure 20-23 Viewing job details



----End

20.4.7 Using Loader to Import Data from a Relational Database to HBase

Scenario

Use Loader to import data from a relational database to HBase.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the HBase tables or phoenix tables that are used during job execution.
- You have obtained the username and password of the relational database.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.
- Before the operation, perform the following steps:

  a. Obtain the JAR package of the relational database driver and save it to the following directory on the active and standby Loader nodes: **${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib**.

  b. Run the following command on the active and standby nodes as user **root** to modify the permission:

  cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

  chown omm:wheel *JAR package name*

  chmod 600 *JAR package name*

c. Log in to FusionInsight Manager. Choose **Cluster** > **Services** > **Loader** > **More** > **Restart Service**. Enter the password of the administrator to restart the Loader service.

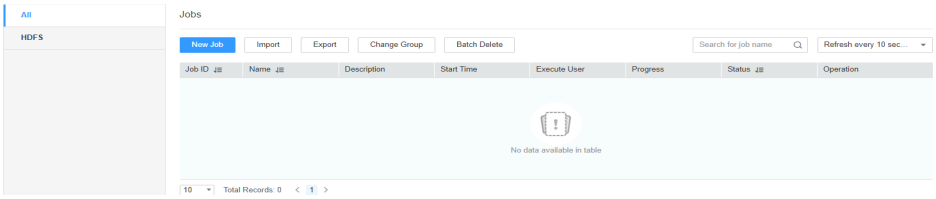
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(*Node name*, Active)**. The Loader web UI is displayed.

Figure 20-24 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-25 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Import**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **generic-jdbc-connector** or a dedicated database connector (oracle-connector, oracle-partition-connector, or mysql-fastpath-connector), set connection parameters, and click **Test** to verify whether the connection is available. When "Test Success" is displayed, click **OK**.

NOTE

- For connection to relational databases, general database connectors (generic-jdbc-connector) or dedicated database connectors (oracle-connector, oracle-partition-connector, and mysql-fastpath-connector) are available. However, compared with general database connectors, dedicated database connectors perform better in data import and export because they are optimized for specific database types.
- When mysql-fastpath-connector is used, the **mysqldump** and **mysqlimport** commands of MySQL must be available on NodeManager nodes, and the MySQL client version to which the two commands belong must be compatible with the MySQL server version. If the two commands are unavailable or the versions are incompatible, install the MySQL client applications and tools following the instructions at **http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**.

Table 20-26 generic-jdbc-connector connection parameters

Parameter	Description	Example Value
Name	Name of a relational database connection	dbName
JDBC Driver Class	Name of a JDBC driver class	oracle.jdbc.driver.OracleDriver
JDBC Connection String	JDBC connection string	jdbc:oracle:thin:@//10.16.0.1:1521/oradb
Username	Username for connecting to the database	omm
Password	Password for connecting to the database	xxxx
JDBC Connection Properties	JDBC connection attribute. Click **Add** to manually add the attribute. • Name: connection attribute name • Value: connection attribute value	• Name: **socketTimeout** • Value: **20**

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set the data source information.

Table 20-27 Parameter description

Parameter	Description	Example Value
Schema Name	Database schema name. This parameter exists in the **Table name** schema.	dbo

Parameter	Description	Example Value
Table Name	Database table name. This parameter exists in the **Table name** schema.	test
SQL Statement	SQL statement for Loader to query data to be imported in **Table SQL statement** mode. The SQL statement requires the query condition **WHERE ${CONDITIONS}**. Without this condition, the SQL statement cannot be run properly. An example SQL statement is as follows: **select * from TABLE WHERE A>B and ${CONDITIONS}**. If **Table column names** is set, the column specified by **Table column names** will replace the column queried in the SQL statement. This parameter cannot be set when **Schema name** or **Table name** is set. **NOTE** You can use macros to define SQL Where statements. For details, see **Using Macro Definitions in Configuration Items** .	select * from test where ${CONDITIONS}
Table Column Names	Table columns whose content is to be imported by Loader. Use commas (,) to separate multiple fields. If the parameter is not set, all the columns are imported and the **Select *** order is used as the column location.	-

Parameter	Description	Example Value
Partition Column Name	Database table column based on which to-be-imported data is determined. This parameter is used for partitioning in a Map job. You are advised to configure the primary key field. **NOTE** • A partition column must have an index. If no index exists, do not specify a partition column. If a partition column without an index is specified, the database server disk I/O will be busy, the access of other services to the database will be affected, and the import will take a long period. • In multiple fields with indexes, select the field that has the most discrete value as the partition column. A partition column that is not discrete may result in load imbalance when multiple MapReduce jobs are imported. • The sorting rules of partition columns must be case-sensitive. Otherwise, data may be lost during data import. • You are not advised to select fields of the float or double type for the partition column. Otherwise, the records containing the minimum and maximum values of the partition column may fail to be imported due to precision issues.	id
Nulls in Partition Column	Indicates whether to process records whose values are null in database table columns. • **true**: Records whose values are null are processed. • **false**: Records whose values are not null are processed.	true
Whether to Specify a Partition Column	Indicates whether to specify a partition column.	true

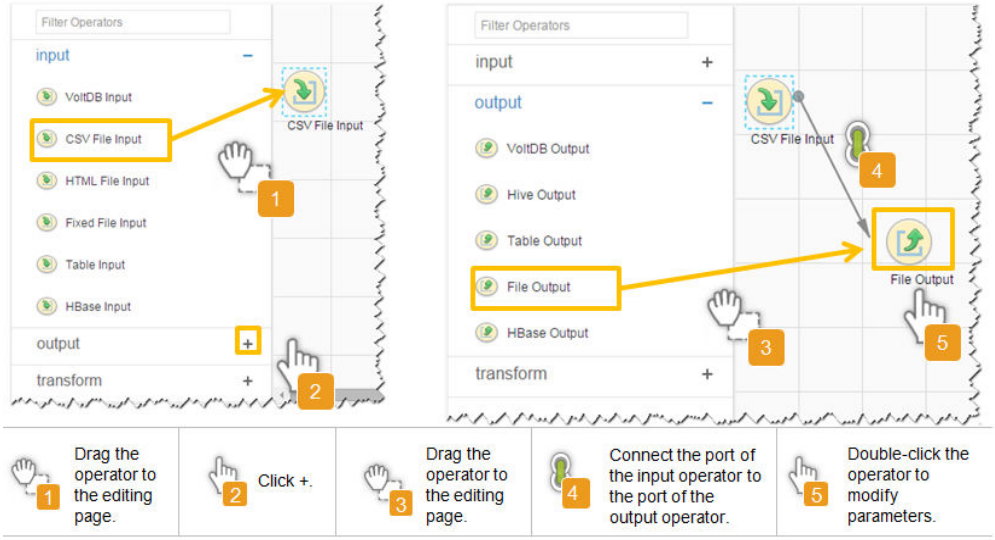
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-28**.

Table 20-28 Input and output parameters of the operator

Input Type	Output Type
Table input	HBase output

Figure 20-26 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set **Storage type** to **HBASE_BULKLOAD** or **HBASE_PUTLIST** based on the actual situation.

Table 20-29 Parameter description

Storage Type	Applicable Scenario	Parameter	Description	Example Value
HBASE_BULKLOAD	Large data volume	HBase Instance	HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.	HBase

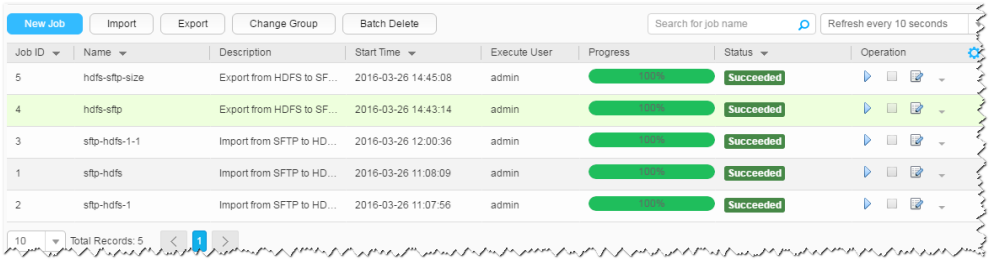
Storage Type	Applicable Scenario	Parameter	Description	Example Value
		Clear data before import	Indicates whether to clear data in the original table before importing data. **True** indicates clearing data and **False** indicates not to clear data. If you do not set this parameter, the original table is not cleared by default.	true
		Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000.	20
		Extractor Size	HBase does not support this parameter. Please set **Extractors**.	-
HBASE_PUTLIST	Small data volume	HBase Instance	HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.	HBase
		Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000.	1000
		Extractor Size	HBase does not support this parameter. Please set **Extractors**.	-

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the Loader web UI. When **Status** is **Succeeded**, the job is complete.

Figure 20-27 Viewing job details



----End

20.4.8 Using Loader to Import Data from a Relational Database to Hive

Scenario

Use Loader to import data from a relational database to Hive.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the Hive tables that are used during job execution.
- You have obtained the username and password of the relational database.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.
- Before the operation, perform the following steps:

  a. Obtain the JAR package of the relational database driver and save it to the following directory on the active and standby Loader nodes: **${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib**.

  b. Run the following command on the active and standby nodes as user **root** to modify the permission:

  cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

  ***chown omm:wheel** JAR package name*

  ***chmod 600** JAR package name*

  c. Log in to FusionInsight Manager. Choose **Cluster** > **Services** > **Loader** > **More** > **Restart Service**. Enter the password of the administrator to restart the Loader service.

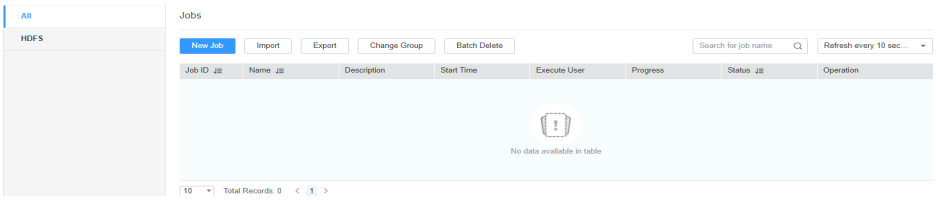
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(*Node name*, Active)**. The Loader web UI is displayed.

Figure 20-28 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-29 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Import**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **generic-jdbc-connector** or a dedicated database connector (oracle-connector, oracle-partition-connector, or mysql-fastpath-connector), set connection parameters, and click **Test** to verify whether the connection is available. When "Test Success" is displayed, click **OK**.

NOTE

- For connection to relational databases, general database connectors (generic-jdbc-connector) or dedicated database connectors (oracle-connector, oracle-partition-connector, and mysql-fastpath-connector) are available. However, compared with general database connectors, dedicated database connectors perform better in data import and export because they are optimized for specific database types.
- When mysql-fastpath-connector is used, the **mysqldump** and **mysqlimport** commands of MySQL must be available on NodeManager nodes, and the MySQL client version to which the two commands belong must be compatible with the MySQL server version. If the two commands are unavailable or the versions are incompatible, install the MySQL client applications and tools following the instructions at **http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**.

Table 20-30 generic-jdbc-connector connection parameters

Parameter	Description	Example Value
Name	Name of a relational database connection	dbName
JDBC Driver Class	Name of a JDBC driver class	oracle.jdbc.driver.OracleDriver
JDBC Connection String	JDBC connection string	jdbc:oracle:thin:@//10.16.0.1:1521/oradb
Username	Username for connecting to the database	omm
Password	Password for connecting to the database	xxxx
JDBC Connection Properties	JDBC connection attribute. Click **Add** to manually add the attribute. • Name: connection attribute name • Value: connection attribute value **NOTICE** If a general connector is used to connect to the MySQL database and there a large amount of data, you need to set **useCursorFetch=true** in the JDBC connection string.	• Name: **socketTimeout** • Value: **20**

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set the data source information.

Table 20-31 Parameter description

Parameter	Description	Example Value
Schema Name	Database schema name. This parameter exists in the **Table name** schema.	dbo
Table Name	Database table name. This parameter exists in the **Table name** schema.	test
SQL Statement	SQL statement for Loader to query data to be imported in **Table SQL statement** mode. The SQL statement requires the query condition **WHERE ${CONDITIONS}**. Without this condition, the SQL statement cannot be run properly. An example SQL statement is as follows: **select * from TABLE WHERE A>B and ${CONDITIONS}**. If **Table column names** is set, the column specified by **Table column names** will replace the column queried in the SQL statement. This parameter cannot be set when **Schema name** or **Table name** is set. **NOTE** You can use macros to define SQL Where statements. For details, see **Using Macro Definitions in Configuration Items** .	select * from test where ${CONDITIONS}
Table Column Names	Table columns whose content is to be imported by Loader. Use commas (,) to separate multiple fields. If the parameter is not set, all the columns are imported and the **Select *** order is used as the column location.	-

Parameter	Description	Example Value
Partition Column Name	Database table column based on which to-be-imported data is determined. This parameter is used for partitioning in a Map job. You are advised to configure the primary key field. **NOTE** • A partition column must have an index. If no index exists, do not specify a partition column. If a partition column without an index is specified, the database server disk I/O will be busy, the access of other services to the database will be affected, and the import will take a long period. • In multiple fields with indexes, select the field that has the most discrete value as the partition column. A partition column that is not discrete may result in load imbalance when multiple MapReduce jobs are imported. • The sorting rules of partition columns must be case-sensitive. Otherwise, data may be lost during data import. • You are not advised to select fields of the float or double type for the partition column. Otherwise, the records containing the minimum and maximum values of the partition column may fail to be imported due to precision issues.	id
Nulls in Partition Column	Indicates whether to process records whose values are null in database table columns. • **true**: Records whose values are null are processed. • **false**: Records whose values are not null are processed.	true
Whether to Specify a Partition Column	Indicates whether to specify a partition column.	true

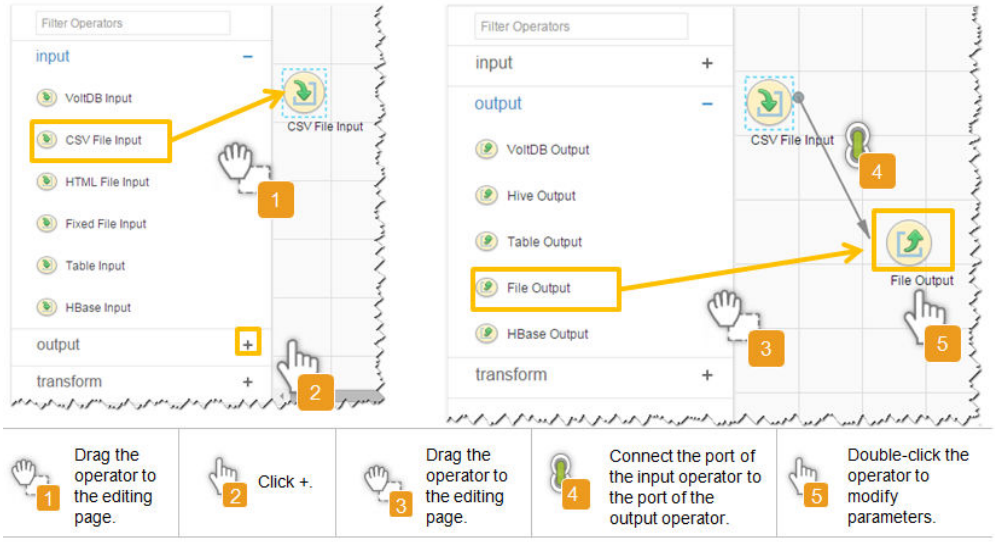
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-32**.

Table 20-32 Input and output parameters of the operator

Input Type	Output Type
Table input	Hive output

Figure 20-30 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set **Storage type** to **HIVE**.

Table 20-33 Parameter description

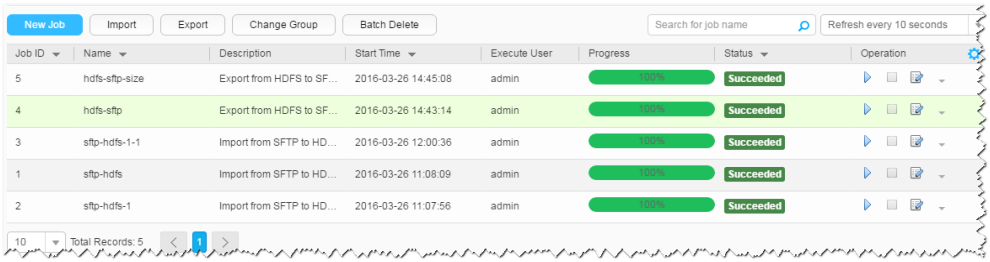
Parameter	Description	Example Value
Output Directory	Directory for storing data imported into Hive. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/opt/tempfile
Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000. You are advised to set the parameter to the maximum number of connections on the SFTP server.	20
Extractor Size	Hive does not support this parameter. Please set **Extractors**.	-

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the Loader web UI. When **Status** is **Succeeded**, the job is complete.

Figure 20-31 Viewing job details



----End

20.4.9 Using Loader to Import Data from HDFS or OBS to HBase

Scenario

Use Loader to import data from HDFS or OBS to HBase.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the HDFS or OBS directories and data involved in job execution.
- You have had the permission to access the HBase tables or phoenix tables that are used during job execution.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- When using Loader to import data from HDFS or OBS, the input paths and input path subdirectories of HDFS or OBS and the name of the files in these directories do not contain any of the special characters /"':;.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.

Procedure

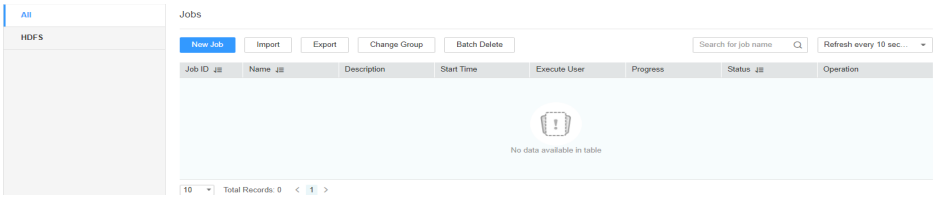
Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.

2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-32 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-33 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Import**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **hdfs-connector**, set connection parameters, and click **Test** to verify whether the connection is available. When "Test Success" is displayed, click **OK**.

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set the data source information.

Table 20-34 Parameter description

Parameter	Description	Example Value
Input Path	Input path of source files in HDFS or OBS **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/user/test
Path Filter	Wildcard for filtering the directories in the input paths of the source files. **Input Path** is not used for filtering. If there are multiple filter conditions, use commas (,) to separate them. If the parameter is empty, the directories are not filtered. The regular expression filtering is not supported.	*
File Filter	Wildcard for filtering the file names of the source files. If there are multiple filter conditions, use commas (,) to separate them. The value cannot be left blank. The regular expression filtering is not supported.	*
Encoding Type	Source file encoding format, for example, UTF-8. This parameter can be set only in text file import.	UTF-8
Suffix	File name extension added to a source file after the source file is imported. If this parameter is empty, no file name extension is added to the source file.	.log

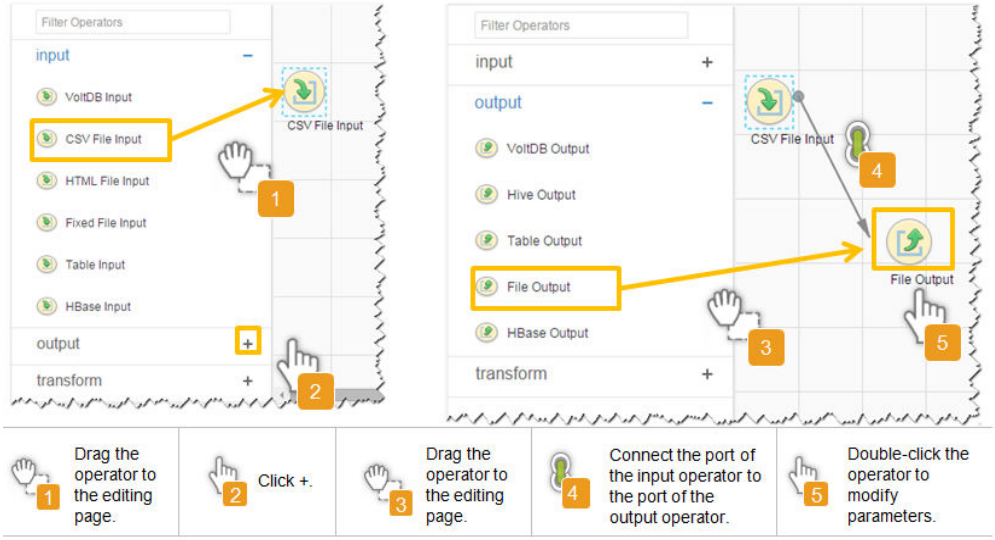
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-35**.

Table 20-35 Input and output parameters of the operator

Input Type	Output Type
CSV File Input	HBase Output
HTML Input	HBase Output
Fixed File Input	HBase Output

Figure 20-34 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set **Storage type** to **HBASE_BULKLOAD** or **HBASE_PUTLIST** based on the actual situation.

Table 20-36 Parameter description

Storage Type	Applicable Scenario	Parameter	Description	Example Value
HBASE_BULKLOAD	Large data volume	HBase Instance	HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.	HBase
		Clear data before import	Indicates whether to clear data in the original table before importing data. **True** indicates clearing data and **False** indicates not to clear data. If you do not set this parameter, the original table is not cleared by default.	true
		Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000.	20

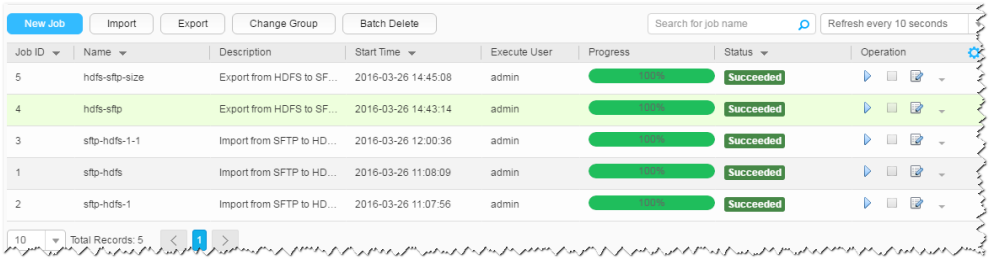
Storage Type	Applicable Scenario	Parameter	Description	Example Value
		Extractor Size	HBase does not support this parameter. Please set **Extractors**.	-
HBASE_PUTLIST	Small data volume	HBase Instance	HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.	HBase
		Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000.	20
		Extractor Size	HBase does not support this parameter. Please set **Extractors**.	-

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the Loader web UI. When **Status** is **Succeeded**, the job is complete.

Figure 20-35 Viewing job details



----End

20.4.10 Using Loader to Import Data from a Relational Database to ClickHouse

Scenario

This section describes how to use Loader to import data from a relational database to ClickHouse using MySQL as an example.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have created a ClickHouse table, and you have operation permissions on the table during job execution.
- You have obtained the user name and password of the MySQL database.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.
- Before the operation, perform the following steps:

  a. Obtain the MySQL client JAR file (for example, **mysqlclient-5.8.1.jar**) from the MySQL database installation path and save it to the following directory on the active and standby Loader nodes:

  ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib.

  b. Obtain the **clickhouse-jdbc-*.jar** file from the ClickHouse installation directory and save it in the **${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib** directory on the active and standby Loader nodes.

  c. Run the following command on the active and standby nodes as user **root** to modify the permission:

  cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

  chown omm:wheel *JAR file name*

  chmod 600 *JAR file name*

  d. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Loader** > **More** > **Restart Service** and enter the administrator password to restart the Loader service.

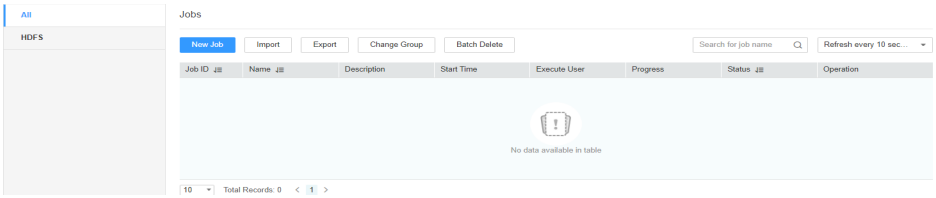
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

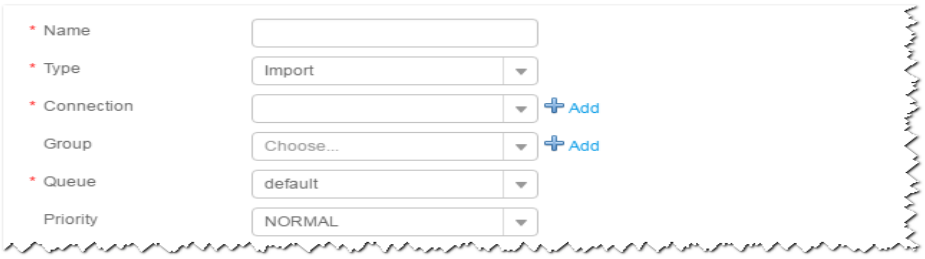
1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-36 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-37 Basic Information



1. Set **Name** to the name of the job.
2. Set **Type** to **Import**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **generic-jdbc-connector** or a dedicated database connector (oracle-connector, oracle-partition-connector, or mysql-fastpath-connector), set connection parameters, and click **Test** to verify whether the connection is available. When "Test Success" is displayed, click **OK**.

NOTE

- For connection to relational databases, general database connectors (generic-jdbc-connector) or dedicated database connectors (oracle-connector, oracle-partition-connector, and mysql-fastpath-connector) are available. However, compared with general database connectors, dedicated database connectors perform better in data import and export because they are optimized for specific database types.
- When mysql-fastpath-connector is used, the **mysqldump** and **mysqlimport** commands of MySQL must be available on NodeManager nodes, and the MySQL client version to which the two commands belong must be compatible with the MySQL server version. If the two commands are unavailable or the versions are incompatible, install the MySQL client applications and tools following the instructions at **http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**.

Table 20-37 generic-jdbc-connector connection parameters

Parameter	Description	Example
Name	Name of a relational database connection	mysql_test
JDBC Driver Class	Name of a JDBC driver class	com.mysql.jdbc.Driver
JDBC Connection String	JDBC connection string	jdbc:mysql://10.254.144.102:3306/test?useUnicode=true&characterEncoding=UTF-8
Username	Username for connecting to the database	root
Password	Password for connecting to the database	xxxx

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, configure the data source information. Currently, only **Table name** is supported.

Table 20-38 Parameter description

Parameter	Description	Example
Schema name	Schema name of the specified database	public
Table name	Table name	test
Table column names	Names of the columns to be imported	id,name
Need partition column	Currently, only the unspecified partition mode is supported.	false

Setting Data Transformation

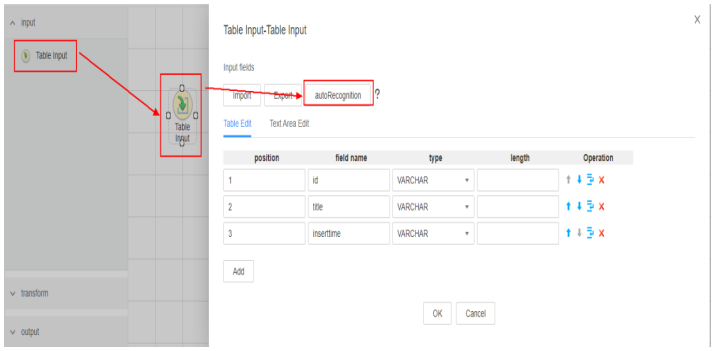
Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-39**.

Table 20-39 Input and output parameters of the operator

Input Type	Output Type
MySQL	ClickHouse

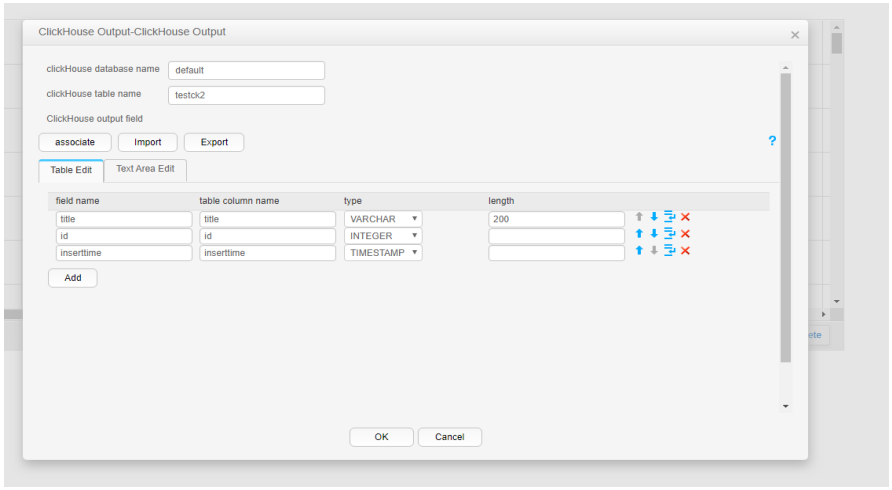
Drag **Table Input** to the grid, double-click **Table Input**, and select **autoRecognition**, as shown in **Figure 20-38**.

Figure 20-38 Operator input



Drag **ClickHouse Output** to the grid, double-click **Table Output**, and select **associate** or manually edit the table to correspond to the input table, as shown in **Figure 20-39**.

Figure 20-39 Operator output



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set **Storage type** to **CLICKHOUSE**.

Table 20-40 Parameter description

Parameter	Description	Example
Storage type	Select **CLICKHOUSE**.	-
ClickHouse instance	Select **ClickHouse**.	-

Parameter	Description	Example
Clear data before import	Select **true** or **false**. **NOTE** If you select **true** and the table to be imported is a ClickHouse distributed table, you need to manually delete the data from the local table corresponding to the ClickHouse distributed table before your import.	true

Step 7 Click **Save and Run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the Loader web UI. When **Status** is **Succeeded**, the job is complete.

Figure 20-40 Viewing job details

Step 9 On the ClickHouse client, check whether the data in the ClickHouse table is the same as that in the MySQL table.

----End

20.4.11 Using Loader to Import Data from HDFS to ClickHouse

Scenario

Use Loader to import data from HDFS to ClickHouse.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the HDFS directories and data involved in job execution.
- You have created a ClickHouse table, and you have operation permissions on the table during job execution.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- When using Loader to import data from HDFS, the input paths and input path subdirectories of HDFS and the name of the files in these directories do not contain any of the special characters **/'':;**.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.

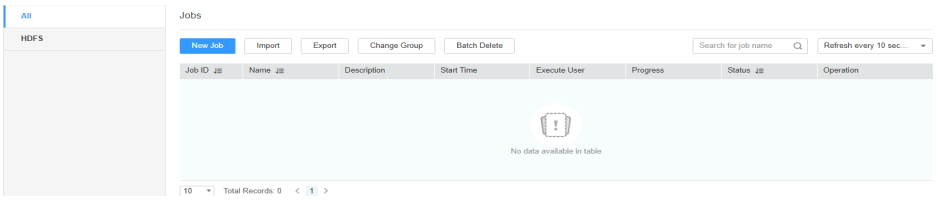
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(*Node name*, Active)**. The Loader web UI is displayed.

Figure 20-41 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-42 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Import**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **hdfs-connector**, set connection parameters, and click **Test** to verify whether the connection is available. When "Test Success" is displayed, click **OK**.

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set the data source information.

Table 20-41 Parameter description

Parameter	Description	Example
Input Path	Input path of source files in HDFS **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/user/test
Path Filter	Wildcard for filtering the directories in the input paths of the source files. **Input Path** is not used for filtering. If there are multiple filter conditions, use commas (**,**) to separate them. If the parameter is empty, the directories are not filtered. The regular expression filtering is not supported.	*
File Filter	Wildcard for filtering the file names of the source files. If there are multiple filter conditions, use commas (**,**) to separate them. The value cannot be left blank. The regular expression filtering is not supported.	*
Encoding Type	Source file encoding format, for example, UTF-8. This parameter can be set only in text file import.	UTF-8
Suffix	File name extension added to a source file after the source file is imported. If this parameter is empty, no file name extension is added to the source file.	.log

Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-42**.

Table 20-42 Input and output parameters of the operator

Input Type	Output Type
CSV File Input	ClickHouse Output

Figure 20-43 Operator operation procedure

Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set **Storage type** to **CLICKHOUSE** based on the actual situation.

Table 20-43 Parameter description

Storage Type	Parameter	Description	Example
CLICKHOUSE	ClickHouse instance	ClickHouse service instance that Loader selects from all available ClickHouse service instances in the cluster. If the selected ClickHouse service instance is not added to the cluster, the ClickHouse job cannot be run properly.	ClickHouse
	Clear data before import	Whether to clear data in the original table before importing data **true** indicates clearing data and **false** indicates not to clear data. If you do not set this parameter, the original table is not cleared by default. **NOTE** If you select **true** and the table to be imported is a ClickHouse distributed table, you need to manually delete the data from the local table corresponding to the ClickHouse distributed table before your import.	false
	Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000.	20
	Extractor size	ClickHouse does not support this parameter. Please set **Extractors**.	-
	Number	Number of Map tasks.	-

Step 7 Click **Save and Run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the Loader web UI. When **Status** is **Succeeded**, the job is complete.

Figure 20-44 Viewing job details

Step 9 On the ClickHouse client, check whether the data in the ClickHouse table is the same as that imported from HDFS.

----End

20.5 Creating a Loader Data Export Job

20.5.1 Using Loader to Export Data from an MRS Cluster

Scenario

This task enables you to export data from MRS to external data sources.

Generally, users can manually manage data import and export jobs on the Loader UI. To use shell scripts to update and run Loader jobs, you must configure the installed Loader client.

Prerequisites

- You have obtained the service user name and password for creating a Loader job.
- You have had the permission to access the HDFS directories, HBase tables, and data involved in job execution.
- You have obtained the user name and password used by an external data source (SFTP server or relational database).
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- When using Loader to export data from HDFS or OBS, the input paths and input path subdirectories of the HDFSor OBS data source and the name of the files in these directories do not contain any of the following special characters: /\"':;,.
- If the job requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.

Procedure

Step 1 Check whether data is exported from Loader to a relational database for the first time.

- If yes, go to **Step 2**.
- If no, go to **Step 3**.

Step 2 Modify the permission on the JAR package of the RDS driver.

1. Obtain the JAR package of the relational database driver and save it to the following directory on the active and standby Loader nodes: **${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib**.
2. Run the following command on the active and standby nodes as user **root** to modify the permission:

   cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

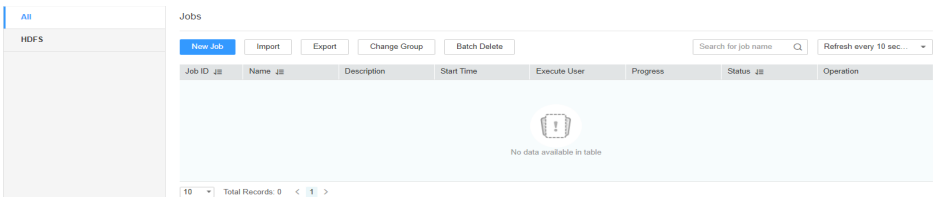
   chown omm:wheel *JAR package name*

   chmod 600 *JAR package name*
3. Log in to FusionInsight Manager. Choose **Cluster** > **Service** > **Loader** > **More** > **Restart**. Enter the password of the administrator to restart the Loader service.

Step 3 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(*Node name*, Active)**. The Loader web UI is displayed.

   Figure 20-45 Loader web UI



Step 4 Create a Loader data import job. Click **New Job**. Select the required job type in **1. Basic Information** and click **Next**.

1. Set **Name** to the job name and **Type** to **Export**.
2. Select a connection for **Connection**. By default, no connection is created. Click **Add** to create a connection, and then click **Test** to test whether the connection is available. Click **OK** when the system displays a message indicates that the test is successful.

Table 20-44 Connection configuration parameters

Connector Type	Parameter	Description
generic-jdbc-connector	JDBC Driver Class	Specifies the name of a JDBC driver class.
	JDBC Connection String	Specifies the JDBC connection string.
	Username	Specifies the username for connecting to the database.
	Password	Specifies the password for connecting to the database.
	JDBC Connection Properties	Specifies JDBC connection attributes. Click **Add** to manually add connection attributes. – **Name**: connection attribute name – **Value**: connection attribute value
hdfs-connector	-	-
oracle-connector	JDBC Connection String	Specifies connection string for a user to connect to the database.
	Username	Specifies the username for connecting to the database.
	Password	Specifies the password for connecting to the database.
	Connection Properties	Specifies connection attributes. Click **Add** to manually add connection attributes. – **Name**: connection attribute name – **Value**: connection attribute value
mysql-fastpath-connector	JDBC Connection String	Specifies the JDBC connection string.
	Username	Specifies the username for connecting to the database.
	Password	Specifies the password for connecting to the database.
	Connection Properties	Specifies connection attributes. Click **Add** to manually add connection attributes. – **Name**: connection attribute name – **Value**: connection attribute value

Connector Type	Parameter	Description
sftp-connector	SFTP Server IP	Specifies the IP address of the SFTP server.
	SFTP Server Port	Specifies the port number of the SFTP server.
	SFTP Username	Specifies the username for accessing the SFTP server.
	SFTP Password	Specifies the password for accessing the SFTP server.
	SFTP Public Key	Specifies public key of the SFTP server.
oracle-partition-connector	JDBC Driver Class	Specifies the name of a Java database connectivity (JDBC) driver class.
	JDBC Connection String	Specifies the JDBC connection string.
	Username	Specifies the username for connecting to the database.
	Password	Specifies the password for connecting to the database.
	Connection Properties	Specifies connection attributes. Click **Add** to manually add connection attributes. – **Name**: connection attribute name – **Value**: connection attribute value

3. Set **Group** to the group to which the job belongs. By default, there is no created group. Click **Add** to create a group and click **OK**.
4. **Queue** indicates that Loader tasks are executed in a specified Yarn queue. The default value is **root.default**, which indicates that the tasks are executed in the **default** queue.
5. Set **Priority** to the priority of Loader tasks in the specified Yarn queue. The options can be **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, or **VERY_HIGH**. The default value is **NORMAL**.

Step 5 In the **2. Input Settings** area, set the data source and click **Next**.

NOTE

When creating or editing a Loader job, you can use macro definitions when configuring parameters such as the SFTP path, HDFS/OBS path, and Where condition of SQL. For details, see **Using Macro Definitions in Configuration Items** .

Table 20-45 List of input configuration parameters

Source File Type	Parameter	Description
HDFS/OBS	Input Directory	Specifies the input path when data is exported from HDFS or OBS.
	Path Filter	Specifies the wildcard for filtering the directories in the input paths of the source files. **Input Directory** is not used in filtering. If there are multiple filter conditions, use commas (,) to separate them. If the value is empty, the directory is not filtered. The regular expression filtering is not supported.
	File Filter	Specifies the wildcard for filtering the file names of the source files. If there are multiple filter conditions, use commas (,) to separate them. The value cannot be left blank. The regular expression filtering is not supported.
	File Type	Specifies the file import type. ● **TEXT_FILE**: imports a text file and stores it as a text file. ● **SEQUENCE_FILE**: imports a text file and stores it as a sequence file. ● **BINARY_FILE**: imports files of any format by using binary streams.
	File Split Type	Specifies whether to split source files by file name or size. The files obtained after the splitting are used as the input files of each map in the MapReduce task for data export.
	Extractors	Specifies the number of maps that are started at the same time in a MapReduce job of a data configuration operation. This parameter cannot be set when **Extractor Size** is set. The value must be less than or equal to 3000.
	Extractor size	Specifies the size of data processed by maps that are started in a MapReduce job of a data configuration operation. The unit is MB. The value must be greater than or equal to 100. The recommended value is **1000**. This parameter cannot be set when **Extractors** is set. When a relational database connector is used, **Extractor size** is unavailable. You need to set **Extractors**.

Source File Type	Parameter	Description
HBASE	HBase Instance	Specifies the HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.
	Quantity	Specifies the number of maps that are started at the same time in a MapReduce job of a data configuration operation. The value must be less than or equal to 3000.
HIVE	Hive instance	Specifies the Hive service instance that Loader selects from all available Hive service instances in the cluster. If the selected Hive service instance is not added to the cluster, the Hive job cannot run properly.
	Quantity	Specifies the number of maps that are started at the same time in a MapReduce job of a data configuration operation. The value must be less than or equal to 3000.
SPARK	Spark instance	Only SparkSQL can access Hive data. Specifies the SparkSQL service instance that Loader selects from all available SparkSQL service instances in the cluster. If the selected Spark service instance is not added to the cluster, the Spark job cannot be run properly.
	Quantity	Specifies the number of maps that are started at the same time in a MapReduce job of a data configuration operation. The value must be less than or equal to 3000.

Step 6 In the **3. Convert** area, set the conversion operations during data transmission.

Check whether source data values in the data operation job created by the Loader can be directly used without conversion, including upper and lower case conversion, cutting, merging, and separation.

- If yes, click **Next**.
- If no, perform **Step 6.1** to **Step 6.4**.

1. No created conversion step exists by default. Drag an example conversion step on the left to the edit box to create a new conversion step.
2. Conversion step types must be selected based on service requirements. A complete conversion process includes the following types:
   a. Input type. Only one conversion step can be added. This parameter is mandatory if the task involves HBase or relational databases.
   b. Conversion type, which is an intermediate conversion step. You can add one or more conversion types or do not add any conversion type.

c. Output type. Only one output type can be added in the last conversion step. This parameter is mandatory if the task involves HBase or relational databases.

Table 20-46 Example list

Type	Description
Input Type	▪ **CSV File Input**: CSV file input step for configuring separators to generate multiple fields. ▪ **Fixed-Width File Input**: Text file input step for configuring the length of characters or bytes to be truncated to generate multiple fields. ▪ **Table Input**: relational data input step for configuring specified columns in the database as input fields. ▪ **HBase Input**: HBase table input step for configuring the column definition of an HBase table to a specified field. ▪ **HTML Input**: HTML web page data input step for obtaining the target data of the HTML web page file to the specified field. ▪ **Hive Input**: Hive table input step for defining columns in a Hive table to specified fields. ▪ **Spark Input**: Spark SQL table input step for defining columns in the SparkSQL table to specified fields. Only Hive data can be stored and accessed.

Type	Description
Conversion type	▪ **Long Integer Time Conversion**: Configure the conversion between a long integer value and a date. ▪ **Null Value Conversion**: Configure a specified value to replace the null value. ▪ **Random Value Conversion**: Configure new value-added fields as random data fields. ▪ **Adding a Constant Field**: Add a constant to directly generate a constant field. ▪ **Concatenation and Conversion**: Concatenate fields, connect generated fields using connection characters, and convert new fields. ▪ **Separator Conversion**: Configure the generated fields to be separated by separators and convert new fields. ▪ **Modulo Conversion**: Configure the generated fields to be converted into new fields through modulo operation. ▪ **Cutting Character String**: Truncate a generated field based on a specified position to generate a new field. ▪ **EL Operation Conversion**: Calculate field values. Currently, the following operators are supported: md5sum, sha1sum, sha256sum, and sha512sum. ▪ **Character String Case Conversion:** Configure the generated fields to be converted to new fields through case conversion. ▪ **Reverse String Conversion**: Reverse the generated fields to generate new fields. ▪ **Character String Space Clearing Conversion**: Configure the generated fields to clear spaces and convert them to new fields. ▪ **Row Filtering Conversion**: Configure logical conditions to filter out rows that contain triggering conditions. ▪ **Update Fields**: Update the value of a specified field when certain conditions are met.

Type	Description
Output type	▪ **File Output**: Configure generated fields to be connected by separators and exported to a file. ▪ **Table Output**: Configure the mapping between output fields and specified columns in the database. ▪ **HBase Output**: Configure the generated fields to the columns of the HBase table. ▪ **Hive Output**: Configure generated fields to a column of a Hive table. ▪ **Spark Output**: Configure generated fields to the columns of SparkSQL tables. Only SparkSQL can access Hive data.

The edit box allows you to perform the following tasks:

- Re-command: Rename an example.
- Edit: Edit the step conversion by referring to **Step 6.3**.
- Delete: Delete an example.

  NOTE

  You can also use the shortcut key Del to delete the file.

3. Click **Edit** to edit the step conversion information and configure fields and data.

   For details about how to set parameters in the step conversion information, see **Loader Operator Help**.

   If the conversion step is incorrectly configured, the source data cannot be converted and become dirty data. The dirty data marking rules are as follows:

   - In any input type step, the number of fields contained in the original data is less than the number of configured fields or the field values in the original data do not match the configured field type.
   - In the **CSV File Input** step, **Validate input field** checks whether the input field matches the value type. If the input field and value type of a line do not match, the line is skipped and becomes dirty data.
   - In the **Fixed Width File Input** step, **Fixed Length** specifies the field splitting length. If the length is greater than the length of the original field value, data splitting fails and the current line becomes dirty data.
   - In the **HBase Input** step, if the HBase table name specified by **HBase Table Name** is incorrect, or no primary key column is configured for Primary Key, all data becomes dirty data.
   - In any conversion step, lines whose conversion fails becomes dirty data. For example, in the **Split Conversion** step, the number of generated fields is less than the number of configured fields, or the original data

cannot be converted to the String type, and the current row becomes dirty data.

- In the **Filter Row Conversion** step, rows filtered by filter criteria become dirty data.
- In the **Modulo Conversion** step, if the original field value is NULL, the current row becomes dirty data.

4. Click **Next**.

Step 7 In the **4. Output Settings** area, set the destination location for saving data and click **Save** to save the job or click **Save and Run** to save and run the job.

Table 20-47 List of Output Configuration Parameters

Data Connection Type	Parameter	Description
sftp-connector	Output Path	Path or name of the export file on an SFTP server. If multiple SFTP server IP addresses are configured for the connector, you can set this parameter to multiple paths or file names separated with semicolons (;). Ensure that the number of input paths or file names is the same as the number of SFTP servers configured for the connector.

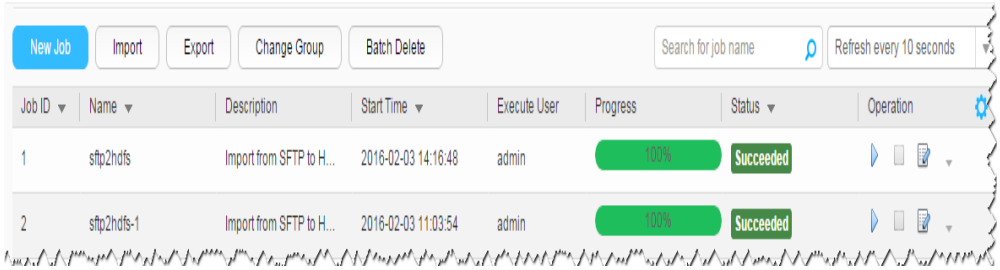
Data Connection Type	Parameter	Description
	Operation	Specifies the action during data import. When all data is to be imported from the input path to the destination path, the data is stored in a temporary directory and then copied from the temporary directory to the destination path. After the data is imported successfully, the data is deleted from the temporary directory. One of the following actions can be taken when duplicate file names exist during data transfer: ● **OVERRIDE**: overrides the old file. ● **RENAME**: renames as new file. For a file without an extension, a string is added to the file name as the extension; for a file with an extension, a string is added to the extension. The string is unique. ● **APPEND**: adds the content of the new file to the end of the old file. This action only adds content regardless of whether the file can be used. For example, a text file can be used after this operation, while a compressed file cannot. ● **IGNORE**: reserves the old file and does not copy the new file. ● **ERROR**: stops the task and reports an error if duplicate file names exist. Transferred files are imported successfully, while files that have duplicate names and files that are not transferred fail to be imported.
	Encode type	Specifies the exported file encoding format, for example, UTF-8. This parameter can be set only in text file export.
	Compression	Indicates whether to enable the compressed transmission function when SFTP is used to export data. **true** indicates that compression is enabled, and **false** indicates that compression is disabled.
hdfs-connector	Output Path	Specifies the output directory or file name of the export file in HDFS or OBS.

Data Connection Type	Parameter	Description
	File Format	Specifies the file export type. • **TEXT_FILE**: imports a text file and stores it as a text file. • **SEQUENCE_FILE**: imports a text file and stores it as a sequence file. • **BINARY_FILE**: imports files of any format by using binary streams.
	Compression codec	Specifies the compression format of files exported to HDFS or OBS. Select a format from the drop-down list. If you select **NONE** or do not set this parameter, data is not compressed.
	User-defined compression format	Name of a user-defined compression format type.
generic-jdbc-connector	Schema name	Specifies the database schema name.
	Table name	Specifies the name of a database table that is used to save the final data of the transmission.
	Temporary table	Specifies the name of a temporary database table that is used to save temporary data during the transmission. The fields in the table must be the same as those in the database specified by **Table name**.
oracle-partition-connector	Schema Name	Specifies the database schema name.
	Table Name	Specifies the name of a database table that is used to save the final data of the transmission.
	Temporary Table	Specifies the name of a temporary database table that is used to save temporary data during the transmission. The fields in the table must be the same as those in the database specified by **Table name**.
oracle-connector	Table Name	Destination table name to store data.
	Column Name	Specifies the name of the column to be written. Columns that are not specified can be set to null or the default value.
mysql-fastpath-connector	Schema Name	Specifies the database schema name.

Data Connection Type	Parameter	Description
	Table Name	Specifies the name of a database table that is used to save the final data of the transmission.
	Temporary Table Name	Name of the temporary table, which is used to store data. After the job is successfully executed, data is transferred to the formal table.

Step 8 On the Loader WebUI page, you can view, start, stop, copy, delete, edit, and view historical information about created jobs.

Figure 20-46 Viewing Loader Jobs



----End

20.5.2 Using Loader to Export Data from HDFS or OBS to an SFTP Server

Scenario

This section describes how to use Loader to export data from HDFS or OBS to an SFTP server.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the HDFS or OBS directories and data involved in job execution.
- You have obtained the username and password of the SFTP server and the user has the write permission of the data export directory on the SFTP server.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- When using Loader to export data from HDFS or OBS, the input paths and input path subdirectories of the HDFS or OBS data source and the name of the files in these directories do not contain any of the following special characters: /\"':;,.

- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.

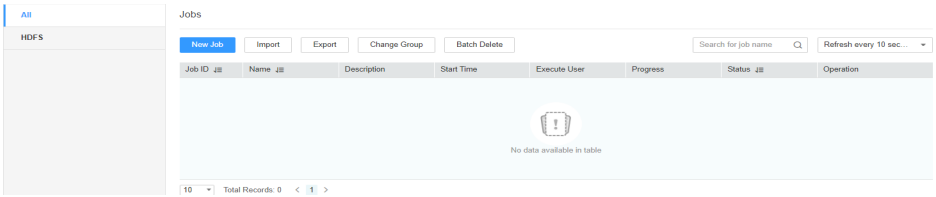
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(*Node name*, Active)**. The Loader web UI is displayed.

Figure 20-47 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-48 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Export**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **sftp-connector**, click **Add**, set connection parameters, and click **Test** to verify whether the connection is available. When "**Test Success**" is displayed, click **OK**. Loader

allows multiple SFTP servers to be configured. Click **Add** to add the configuration information of multiple SFTP servers.

Table 20-48 Connection parameters

Parameter	Description	Example Value
Name	Specifies the name of the SFTP server connection.	sftpName
SFTP Server IP	Specifies the IP address of the SFTP server.	10.16.0.1
SFTP Server Port	Specifies the port number of the SFTP server.	22
SFTP Username	Specifies the user name for accessing the SFTP server.	root
SFTP Password	Specifies the password for accessing the SFTP server.	xxxx
SFTP Public Key	Specifies public key of the SFTP server.	OdDt/yn...etM

NOTE

When multiple SFTP servers are configured, the data of HDFSor OBS will be divided into multiple parts and exported to the SFTP servers randomly.

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set **Source type** to **HDFS**.

Table 20-49 Data source parameters

Parameter	Description	Example Value
Input directory	Specifies the input path when data is exported from HDFS or OBS. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/user/test

Parameter	Description	Example Value
Path filter	Specifies the wildcard for filtering the directories in the input paths of the source files. **Input directory** is not used in filtering. If there are multiple filter conditions, use commas (**,**) to separate them. If the parameter is empty, the directory is not filtered. The regular expression filtering is not supported. ● **?** matches a single character. ● ***** indicates multiple characters. ● Adding **^** before the condition indicates negated filtering, that is, file filtering.	*
File filter	Specifies the wildcard for filtering the file names of the source files. If there are multiple filter conditions, use commas (**,**) to separate them. The value cannot be left blank. The regular expression filtering is not supported. ● **?** matches a single character. ● ***** indicates multiple characters. ● Adding **^** before the condition indicates negated filtering, that is, file filtering.	*
File Type	Specifies the file import type. ● **TEXT_FILE**: imports a text file and stores it as a text file. ● **SEQUENCE_FILE**: imports a text file and stores it as a **sequence file**. ● **BINARY_FILE**: imports files of any format by using binary streams but not to process the files. **NOTE** When the file import type to **TEXT_FILE** or **SEQUENCE_FILE**, Loader automatically selects a decompression method based on the file name extension to decompress a file.	TEXT_FILE

Parameter	Description	Example Value
File Split Type	Indicates whether to split source files by file name or size. The files obtained after the splitting are used as the input files of each map in the MapReduce task for data export. ● **FILE**: indicates that the source file is split by file. That is, each map processes one or multiple complete files, the same source file cannot be allocated to different maps, and the source file directory structure is retained after data import. ● **SIZE**: indicates that the source file is split by size. That is, each map processes input files of a certain size, and a source file can be divided and processed by multiple maps. After data is stored in the output directory, the number of saved files is the same as the number of maps. The file name format is **import_part_xxxx**, where **xxxx** is a unique random number generated by the system.	FILE
Extractors	Specifies the number of maps that are started at the same time in a MapReduce job of a data configuration operation. This parameter cannot be set when **Extractor Size** is set. The value must be less than or equal to 3000. You are advised to set the parameter to the number of CPU cores on the SFTP server. **NOTE** To improve the data import speed, ensure that the following conditions are met: ● Each map connection is equivalent to a client connection. Therefore, you must ensure that the maximum number of connections of the SFTP server is greater than the number of maps. ● Ensure that the disk I/O or network bandwidth on the SFTP server does not reach the upper limit.	20
Extractor size	Specifies the size of data processed by maps that are started in a MapReduce job of a data configuration operation. The unit is MB. The value must be greater than or equal to 100. The recommended value is 1000. This parameter cannot be set when **Extractors** is set.	-

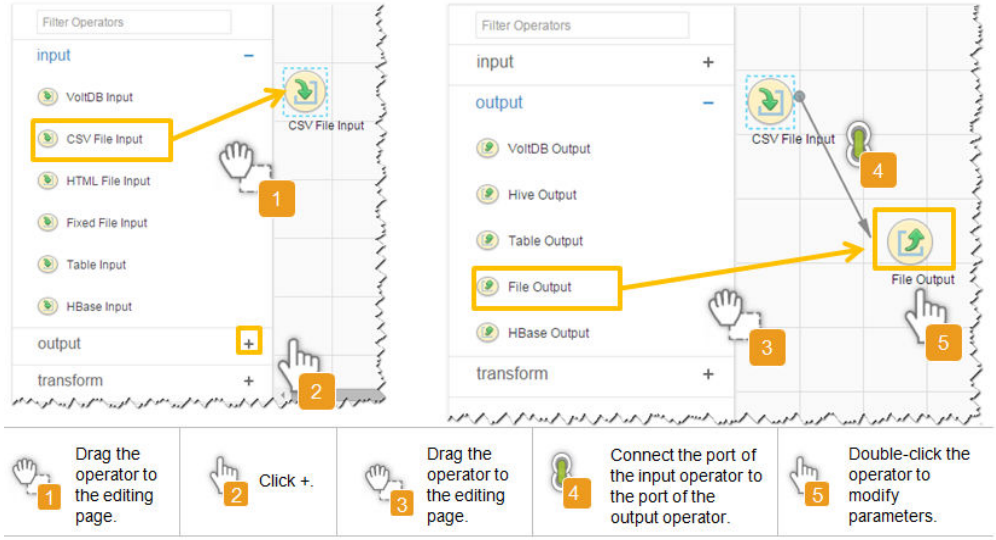
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-50**.

Table 20-50 Setting the input and output parameters of the operator

Input Type	Export Type
CSV file input	File output
HTML input	File output
Fixed-width file input	File output

Figure 20-49 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set the data storage mode.

Table 20-51 Parameter description

Parameter	Description	Example Value
Output path	Specifies the path or file name of the exported file on an SFTP server. If multiple SFTP server IP addresses are configured for the connector, you can set this parameter to multiple paths or file names separated with semicolons (;). Ensure that the number of paths or file names is the same as the number of SFTP servers configured for the connector. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/opt/tempfile
Operation	Specifies the action during data import. When all data is to be imported from the input path to the destination path, the data is stored in a temporary directory and then copied from the temporary directory to the destination path. After the data is imported successfully, the data is deleted from the temporary directory. One of the following actions can be taken when duplicate file names exist during data transfer: • **OVERRIDE**: overrides the old file. • **RENAME**: renames as new file. For a file without an extension, a string is added to the file name as the extension; for a file with an extension, a string is added to the extension. The string is unique. • **APPEND**: adds the content of the new file to the end of the old file. This action only adds content regardless of whether the file can be used. For example, a text file can be used after this operation, while a compressed file cannot. • **IGNORE**: reserves the old file and does not copy the new file. • **ERROR**: stops the task and reports an error if duplicate file names exist. Transferred files are imported successfully, while files that have duplicate names and files that are not transferred fail to be imported.	OVERRIDE
Encode type	Specifies the exported file encoding format, for example, UTF-8. This parameter can be set only in text file export.	UTF-8

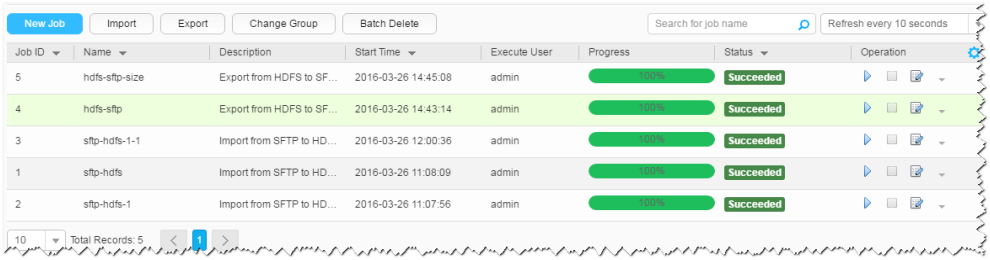
Parameter	Description	Example Value
Compression	Indicates whether to enable the compressed transmission function when SFTP is used to export data. • The value **true** indicates that compression is enabled. • The value **false** indicates that compression is disabled.	true

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the **Loader WebUI**. When **Status** is **Succeeded**, the job is complete.

Figure 20-50 Viewing a job



----End

20.5.3 Using Loader to Export Data from HBase to an SFTP Server

Scenario

Use Loader to export data from HBase to an SFTP server.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the HBase tables or phoenix tables that are used during job execution.
- You have obtained the username and password of the SFTP server and the user has the write permission of the data export directory on the SFTP server.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.

- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.

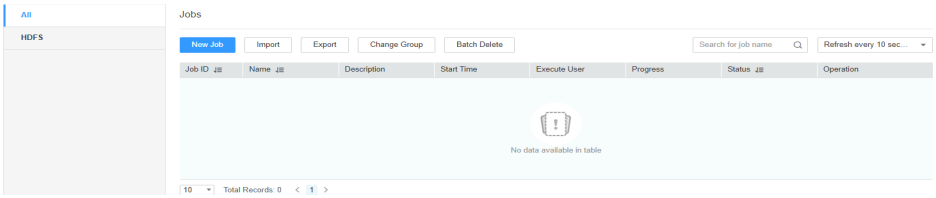
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(*Node name*, Active)**. The Loader web UI is displayed.

Figure 20-51 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-52 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Export**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **sftp-connector**, click **Add**, set connection parameters, and click **Test** to verify whether the connection is available. When "**Test Success**" is displayed, click **OK**. Loader allows multiple SFTP servers to be configured. Click **Add** to add the configuration information of multiple SFTP servers.

Table 20-52 Connection parameters

Parameter	Description	Example Value
Name	Specifies the name of the SFTP server connection.	sftpName
SFTP server IP	Specifies the IP address of the SFTP server.	10.16.0.1
SFTP server port	Specifies the port number of the SFTP server.	22
SFTP username	Specifies the user name for accessing the SFTP server.	root
SFTP password	Specifies the password for accessing the SFTP server.	xxxx
SFTP public key	Specifies public key of the SFTP server.	OdDt/yn...etM

NOTE

When multiple SFTP servers are configured, the data of HBase tables or phoenix tables will be divided into multiple parts and saved to the SFTP servers randomly.

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set **Source type** to **HBASE**.

Table 20-53 Data source parameters

Parameter	Description	Example Value
HBase instance	Specifies the HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.	HBase
Quantity	Specifies the number of maps that are started at the same time in a MapReduce job of a data configuration operation. The value must be less than or equal to 3000. You are advised to set the parameter to the maximum number of connections on the SFTP server.	20

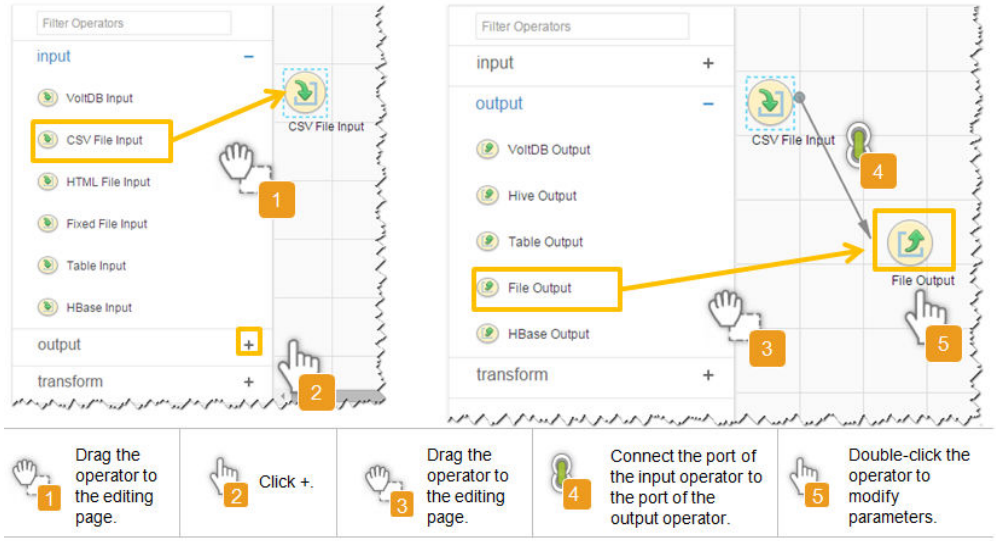
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-54**.

Table 20-54 Setting the input and output parameters of the operator

Input Type	Export Type
HBase input	File output

Figure 20-53 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set the data storage mode.

Table 20-55 Parameter description

Parameter	Description	Example Value
Output path	Specifies the path or file name of the exported file on an SFTP server. If multiple SFTP server IP addresses are configured for the connector, you can set this parameter to multiple paths or file names separated with semicolons (;). Ensure that the number of paths or file names is the same as the number of SFTP servers configured for the connector. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/opt/tempfile

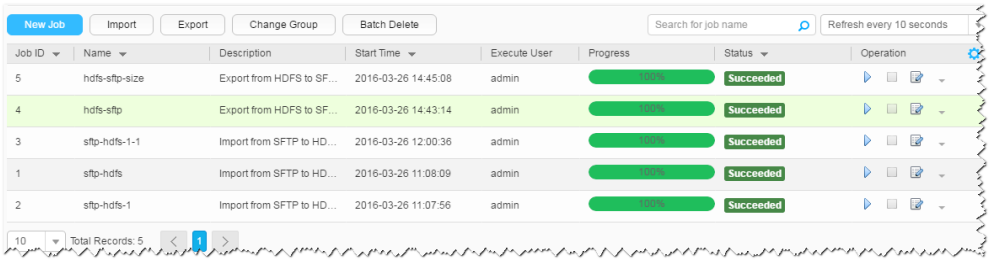
Parameter	Description	Example Value
Operation	Specifies the action during data import. When all data is to be imported from the input path to the destination path, the data is stored in a temporary directory and then copied from the temporary directory to the destination path. After the data is imported successfully, the data is deleted from the temporary directory. One of the following actions can be taken when duplicate file names exist during data transfer: ● **OVERRIDE**: overrides the old file. ● **RENAME**: renames as new file. For a file without an extension, a string is added to the file name as the extension; for a file with an extension, a string is added to the extension. The string is unique. ● **APPEND**: adds the content of the new file to the end of the old file. This action only adds content regardless of whether the file can be used. For example, a text file can be used after this operation, while a compressed file cannot. ● **IGNORE**: reserves the old file and does not copy the new file. ● **ERROR**: stops the task and reports an error if duplicate file names exist. Transferred files are imported successfully, while files that have duplicate names and files that are not transferred fail to be imported.	OVERRIDE
Encode type	Specifies the exported file encoding format, for example, UTF-8. This parameter can be set only in text file export.	UTF-8
Compression	Indicates whether to enable the compressed transmission function when SFTP is used to export data. ● The value **true** indicates that compression is enabled. ● The value **false** indicates that compression is disabled.	true

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the **Loader WebUI**. When **Status** is **Succeeded**, the job is complete.

Figure 20-54 Viewing a job



----End

20.5.4 Using Loader to Export Data from Hive to an SFTP Server

Scenario

Use Loader to export data from Hive to an SFTP server.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the Hive table specified in the job.
- You have obtained the username and password of the SFTP server and the user has the write permission of the data export directory on the SFTP server.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.

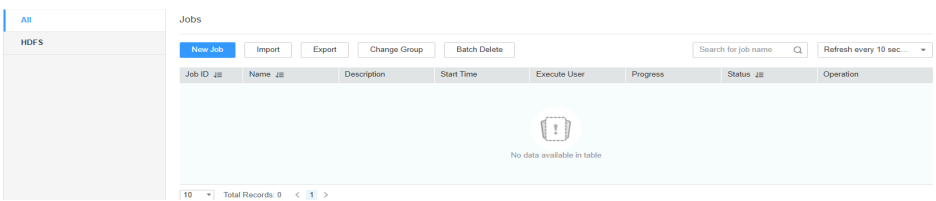
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-55 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-56 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Export**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **sftp-connector**, click **Add**, set connection parameters, and click **Test** to verify whether the connection is available. When "**Test Success**" is displayed, click **OK**. Loader allows multiple SFTP servers to be configured. Click **Add** to add the configuration information of multiple SFTP servers.

Table 20-56 Connection parameters

Parameter	Description	Example Value
Name	Specifies the name of the SFTP server connection.	sftpName
SFTP server IP	Specifies the IP address of the SFTP server.	10.16.0.1
SFTP server port	Specifies the port number of the SFTP server.	22
SFTP username	Specifies the user name for accessing the SFTP server.	root
SFTP password	Specifies the password for accessing the SFTP server.	xxxx
SFTP public key	Specifies public key of the SFTP server.	OdDt/yn...etM

NOTE

When multiple SFTP servers are configured, the data of Hive tables will be divided into multiple parts and saved to the SFTP servers randomly.

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set **Source type** to **HIVE**.

Table 20-57 Data source parameters

Parameter	Description	Example Value
Hive instance	Specifies the Hive service instance that Loader selects from all available Hive service instances in the cluster. If the selected Hive service instance is not added to the cluster, the Hive job cannot run properly.	hive
Quantity	Specifies the number of maps that are started at the same time in a MapReduce job of a data configuration operation. The value must be less than or equal to 3000. You are advised to set the parameter to the maximum number of connections on the SFTP server.	20

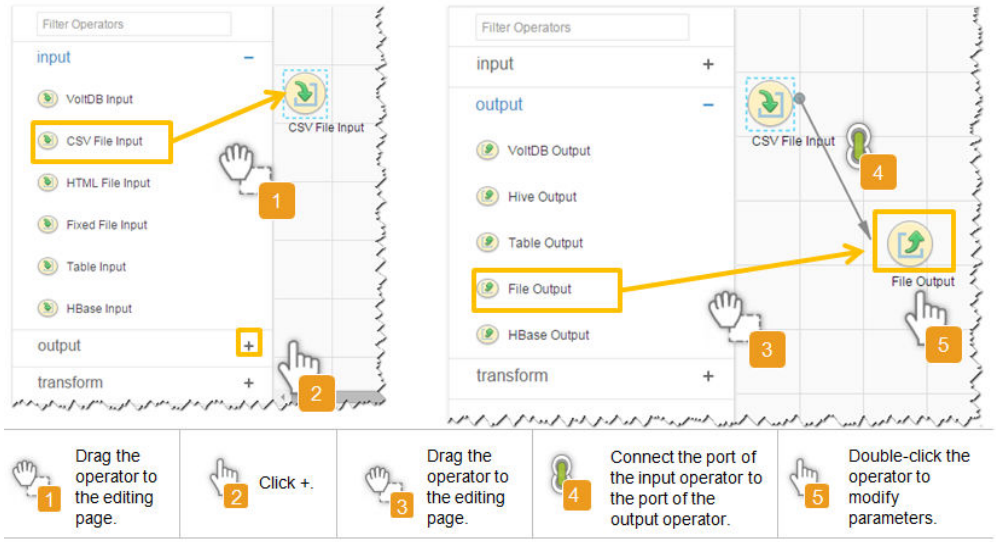
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-58**.

Table 20-58 Setting the input and output parameters of the operator

Input Type	Export Type
Hive input	File output

Figure 20-57 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set the data storage mode.

Table 20-59 Parameter description

Parameter	Description	Example Value
Output path	Specifies the path or file name of the exported file on an SFTP server. If multiple SFTP server IP addresses are configured for the connector, you can set this parameter to multiple paths or file names separated with semicolons (;). Ensure that the number of paths or file names is the same as the number of SFTP servers configured for the connector. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/opt/tempfile

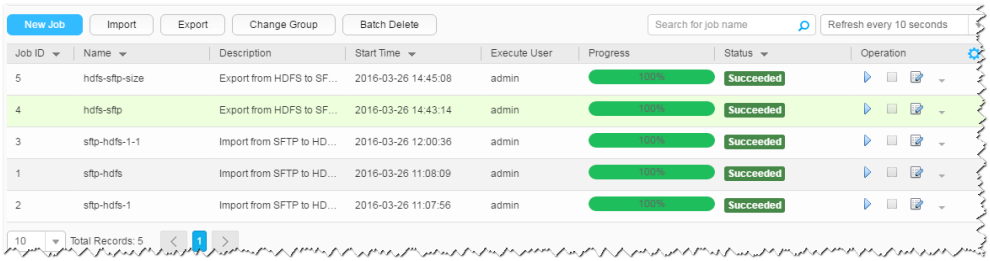
Parameter	Description	Example Value
Operation	Specifies the action during data import. When all data is to be imported from the input path to the destination path, the data is stored in a temporary directory and then copied from the temporary directory to the destination path. After the data is imported successfully, the data is deleted from the temporary directory. One of the following actions can be taken when duplicate file names exist during data transfer: ● **OVERRIDE**: overrides the old file. ● **RENAME**: renames as new file. For a file without an extension, a string is added to the file name as the extension; for a file with an extension, a string is added to the extension. The string is unique. ● **APPEND**: adds the content of the new file to the end of the old file. This action only adds content regardless of whether the file can be used. For example, a text file can be used after this operation, while a compressed file cannot. ● **IGNORE**: reserves the old file and does not copy the new file. ● **ERROR**: stops the task and reports an error if duplicate file names exist. Transferred files are imported successfully, while files that have duplicate names and files that are not transferred fail to be imported.	OVERRIDE
Encode type	Specifies the exported file encoding format, for example, UTF-8. This parameter can be set only in text file export.	UTF-8
Compression	Indicates whether to enable the compressed transmission function when SFTP is used to export data. ● The value **true** indicates that compression is enabled. ● The value **false** indicates that compression is disabled.	true

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the **Loader WebUI**. When **Status** is **Succeeded**, the job is complete.

Figure 20-58 Viewing a job



----End

20.5.5 Using Loader to Export Data from HDFS or OBS to a Relational Database

Scenario

This section describes how to use Loader to export data from HDFS or OBS to a relational database.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the HDFS or OBS directories and data involved in job execution.
- You have obtained the username and password of the relational database.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.
- Before the operation, perform the following steps:
  a. Obtain the JAR package of the relational database driver and save it to the following directory on the active and standby Loader nodes: **${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib**.
  b. Run the following command on the active and standby nodes as user root to modify the permission:

     cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

     chown omm:wheel *JAR package name*

     chmod 600 *JAR package name*
  c. Log in to FusionInsight Manager. Choose **Cluster** > **Service** > **Loader** > **More** > **Restart**. Enter the password of the administrator to restart the Loader service.

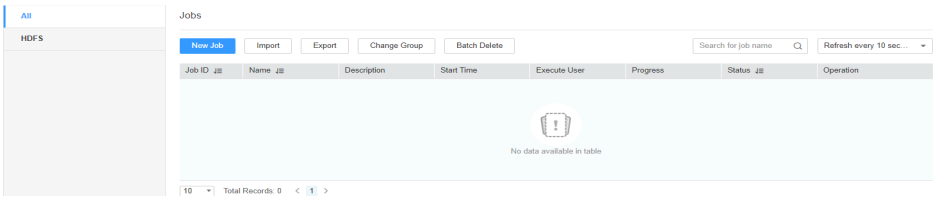
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(*Node name*, Active)**. The Loader web UI is displayed.

Figure 20-59 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-60 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Export**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **generic-jdbc-connector** or dedicated database connector (oracle-connector, oracle-partition-connector or mysql-fastpath-connector), set connection parameters, and click **Test** to verify whether the connection is available. When "**Test Success**" is displayed, click **OK**.

NOTE

- For connection to relational databases, general database connectors (generic-jdbc-connector) or dedicated database connectors (oracle-connector, oracle-partition-connector, and mysql-fastpath-connector) are available. However, compared with general database connectors, dedicated database connectors perform better in data import and export because they are optimized for specific database types.
- When **mysql-fastpath-connector** is used, the **mysqldump** and **mysqlimport** commands of MySQL must be available on NodeManagers, and the MySQL client version to which the two commands belong must be compatible with the MySQL server version. If the two commands are unavailable or the versions are incompatible, see **http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**. Install the MySQL client applications and tools.

Table 20-60 generic-jdbc-connector connection parameters

Parameter	Description	Example Value
Name	Specifies the name of a relational database connection.	dbName
JDBC Driver Class	Specifies the name of a Java database connectivity (JDBC) driver class.	oracle.jdbc.driver.OracleDriver
JDBC Connection String	Specifies the JDBC connection string.	jdbc:oracle:thin:@//10.16.0.1:1521/oradb
Username	Specifies the username for connecting to the database.	omm
Password	Specifies the password for connecting to the database.	xxxx
JDBC Connection Properties	JDBC connection attribute. Click **Add** to manually add the attribute. • Name: connection attribute name • Value: connection attribute value	• Name: socketTimeout • Value: 20

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set **Source type** to **HDFS**.

Table 20-61 Data source parameters

Parameter	Description	Example Value
Input directory	Specifies the input path when data is exported from HDFS or OBS. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/user/test
Path filter	Specifies the wildcard for filtering the directories in the input paths of the source files. **Input directory** is not used in filtering. If there are multiple filter conditions, use commas (,) to separate them. If the parameter is empty, the directory is not filtered. The regular expression filtering is not supported. ● **?** matches a single character. ● ***** indicates multiple characters. ● Adding ^ before the condition indicates negated filtering, that is, file filtering.	*
File filter	Specifies the wildcard for filtering the file names of the source files. If there are multiple filter conditions, use commas (,) to separate them. The value cannot be left blank. The regular expression filtering is not supported. ● **?** matches a single character. ● ***** indicates multiple characters. ● Adding ^ before the condition indicates negated filtering, that is, file filtering.	*
File Type	Specifies the file import type. ● **TEXT_FILE**: imports a text file and stores it as a text file. ● **SEQUENCE_FILE**: imports a text file and stores it as a sequence file. ● **BINARY_FILE**: imports files of any format by using binary streams but not to process the files. **NOTE** When the file import type to **TEXT_FILE** or **SEQUENCE_FILE**, Loader automatically selects a decompression method based on the file name extension to decompress a file.	TEXT_FILE

Parameter	Description	Example Value
File split type	Indicates whether to split source files by file name or size. The files obtained after the splitting are used as the input files of each map in the MapReduce task for data export. • **FILE**: indicates that the source file is split by file. That is, each map processes one or multiple complete files, the same source file cannot be allocated to different maps, and the source file directory structure is retained after data import. • **SIZE**: indicates that the source file is split by size. That is, each map processes input files of a certain size, and a source file can be divided and processed by multiple maps. After data is stored in the output directory, the number of saved files is the same as the number of maps. The file name format is **import_part_xxxx**, where **xxxx** is a unique random number generated by the system.	FILE
Extractors	Specifies the number of maps that are started at the same time in a MapReduce job of a data configuration operation. This parameter cannot be set when **Extractor size** is set. The value must be less than or equal to 3000.	20
Extractor size	Specifies the size of data processed by maps that are started in a MapReduce job of a data configuration operation. The unit is MB. The value must be greater than or equal to 100. The recommended value is 1000. This parameter cannot be set when **Extractors** is set. When a relational database connector is used, **Extractor size** is unavailable. You need to set **Extractors**.	-

Setting Data Transformation

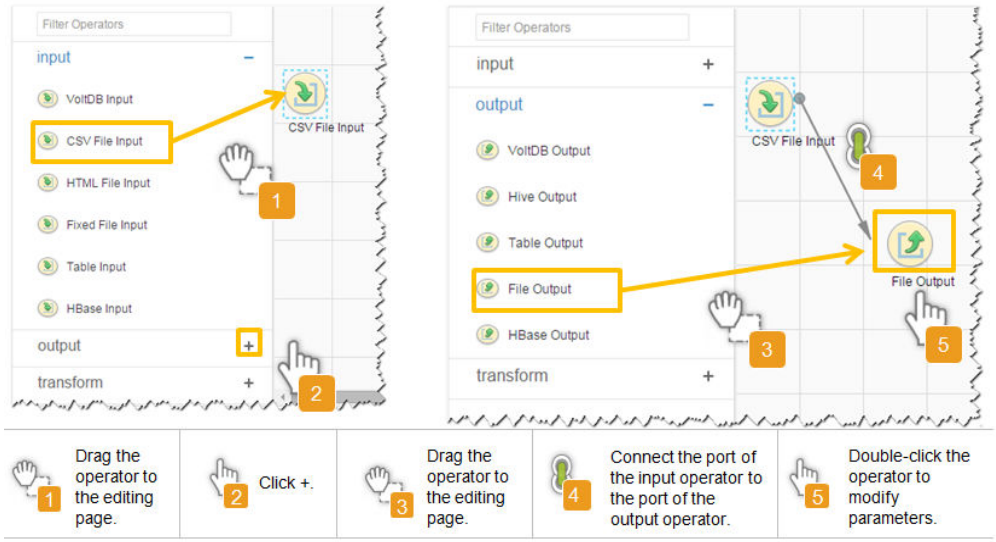
Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-62**.

Table 20-62 Setting the input and output parameters of the operator

Input Type	Export Type
CSV file input	Table output

Input Type	Export Type
HTML Input	Table output
Fixed-width file input	Table output

Figure 20-61 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set the data storage mode.

Table 20-63 Parameter description

Parameter	Description	Example Value
Schema name	Specifies the database schema name.	dbo
Table Name	Specifies the name of a database table that is used to save the final data of the transmission. **NOTE** Table names can be defined using macros. For details, see **Using Macro Definitions in Configuration Items** .	test

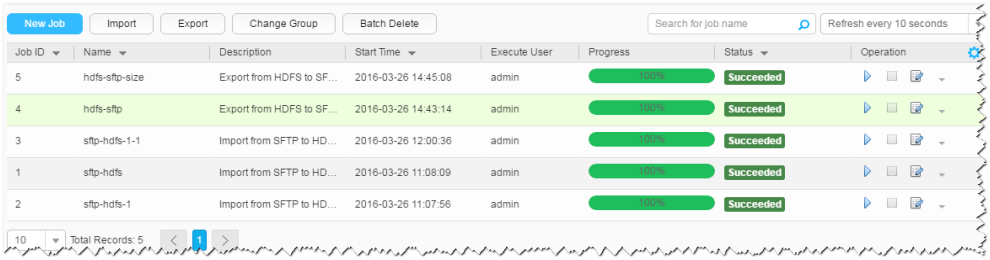
Parameter	Description	Example Value
Temporary table	Specifies the name of a temporary database table that is used to save temporary data during the transmission. The fields in the table must be the same as those in the database specified by **Table name**. **NOTE** A temporary table is used to prevent dirty data from being generated in the destination table when data is exported to the database. Data is migrated from the temporary table to the destination table only after all data is successfully written to the temporary table. Using temporary tables increases the job execution time.	tmp_test

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the **Loader WebUI**. When **Status** is **Succeeded**, the job is complete.

Figure 20-62 Viewing a job



----End

20.5.6 Using Loader to Export Data from HDFS to MOTService

Scenario

This section applies to MRS 3.3.0 or later.

In MOTService, tables need to be updated based on the data version field in the tables. Tables outside MOTService do not support Upsert statements. You can use Loader to export these tables from HDFS to MOTService to update their data in batches.

Prerequisites

- You have obtained the username and password of the relational database.
- The input data must be in CSV format.

- You have created a human-machine user, for example, **Loaderuser**, and added the user to user groups **hive** (primary) and **hadoop**, and associated the user with the **Manager_administrator** role on FusionInsight Manager.

Procedure

Make preparations.

Step 1 Log in to the node where RTDServer is installed as user **root** and obtain the driver JAR file corresponding to the relational database, for example, **opengaussjdbc-V500R002C00.jar** in this scenario.

cd ${BIGDATA_HOME}/FusionInsight_FARMER_RTD_*/install/FusionInsight-RTD-*/RTD/rtdservice/WEB-INF/lib

Step 2 Save the JAR file obtained in **Step 1** to the **${BIGDATA_HOME}/FusionInsight_Porter_*/install/FusionInsight-Sqoop-*/FusionInsight-Sqoop-*/server/webapps/loader/WEB-INF/ext-lib** directory on the active and standby Loader nodes.

Step 3 Run the following command on the active and standby nodes as user **root** to modify the permission on the JAR file:

cd ${BIGDATA_HOME}/FusionInsight_Porter_*/install/FusionInsight-Sqoop-*/FusionInsight-Sqoop-*/server/webapps/loader/WEB-INF/ext-lib

chown omm:wheel *JAR file name*

chmod 600 *JAR file name*

Step 4 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Loader**. On the **Dashboard** tab page that is displayed, click **More** and select **Restart Service**. In the displayed dialog box, enter the administrator password to restart the Loader service.

Step 5 Create a version control table in MOTService and add specific fields to the table for version control. If there is such a table, you do not need to create one. All MOT jobs (full or incremental) share the same table. The reference commands are as follows:

CREATE TABLE T_RTD_TBL_CUR_VER_INFO (

TBL_NAME varchar NOT NULL,

CUR_VER_FLAG tinyint DEFAULT '0' NOT NULL,

CONSTRAINT PK_T_RTD_TBL_CUR_VER_INFO PRIMARY KEY (TBL_NAME)

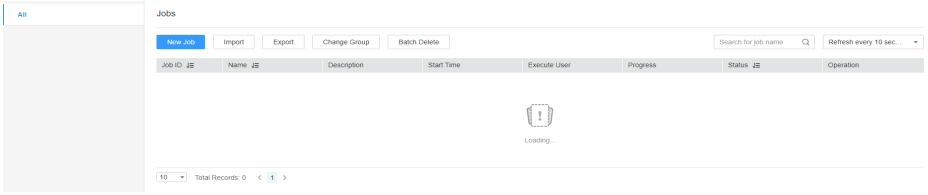
);

Configure basic job information.

Step 6 Access the Loader web UI.

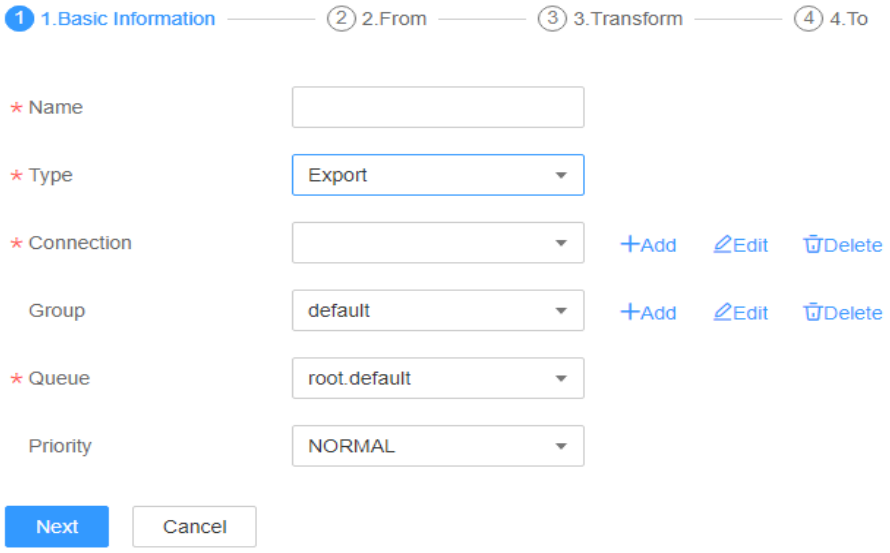
1. Log in to FusionInsight Manager.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-63 Loader web UI



Step 7 Click **New Job**. Configure basic job parameters on the **Basic Information** page displayed.

Figure 20-64 Basic Information page



1. Enter a job name in **Name**.
2. Set **Type** to **Export**.
3. Set **Group** to the group to which the job belongs. There is no group created by default. Click **Add**, enter the group name, and click **OK**.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 8 In the **Connection** area, click **Add** to create a connection, set **Connector** to **generic-jdbc-connector**, set connection parameters, and click **Test** to verify whether the connection is available. When "Test Success" is displayed, click **OK**.

Table 20-64 generic-jdbc-connector connection parameters

Parameter	Description	Example Value
Name	Name of a relational database connection	dbName

Parameter	Description	Example Value
JDBC Driver Class	Name of a JDBC driver class	com.huawei.opengauss.jdbc.Driver
JDBC Connection String	JDBC connection string, in the following format: **jdbc:opengauss://** *Database IP address*:*Database port number*/*Database name*	jdbc:opengauss://10.10.10.10:15400/test
Username	Username for connecting to the database	omm
Password	Password for connecting to the database	xxxx

Parameter	Description	Example Value
JDBC Connection Properties	JDBC connection attribute. Click **Add** to manually add the attribute. • **Name**: connection attribute name • **Value**: connection attribute value	• Name: **socketTimeout** • Value: **20** **NOTE** Log in to FusionInsight Manager and choose **Cluster** > **Services** > **MOTService**. Click **Configurations** then **All Configurations**, and search for the **REQUIRE_SSL** parameter. If the parameter value is **true**, add a JDBC connection attribute whose name is **ssl.enable** and value is **true**. Otherwise, you do not need to add this connection attribute.

Configure data source information.

Step 9 Click **Next**. On the **From** page displayed, set **Source type** to **HDFS**.

Table 20-65 Data source parameters

Parameter	Description	Example Value
Input directory	Input path when data is exported from HDFS **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/user/test
Path filter	Wildcard for filtering the directories in the input paths of the source files. **Input directory** is not used for filtering. Use commas (,) to separate multiple filter criteria. If this parameter is left blank, directories are not filtered. Regular expression filtering is not supported. • **?** matches a single character. • ***** indicates multiple characters. • Adding ^ before the condition indicates negated filtering, that is, file filtering.	*
File filter	Wildcard for filtering the file names of the source files. Use commas (,) to separate multiple filter criteria. This parameter cannot be left blank. Regular expression filtering is not supported. • **?** matches a single character. • ***** indicates multiple characters. • Adding ^ before the condition indicates negated filtering, that is, file filtering.	*
File type	File import type. The options are as follows: • **TEXT_FILE**: imports a text file and saves it as a text file. • **SEQUENCE_FILE**: imports a text file and saves it as a sequence file. • **BINARY_FILE**: imports files of any format using binary streams but not to process the files. **NOTE** When the file import type is set to **TEXT_FILE** or **SEQUENCE_FILE**, Loader automatically selects a decompression method based on the file name extension to decompress a file.	TEXT_FILE

Parameter	Description	Example Value
File split type	Whether to split source files by file name or size. The files obtained after the splitting are used as the input files of each Map in the MapReduce task for data export. • **FILE**: indicates that the source file is split by file. That is, each Map processes one or multiple complete files, the same source file cannot be allocated to different Maps, and the source file directory structure is retained after data import. • **SIZE**: indicates that the source file is split by size. That is, each Map processes input files of a certain size, and a source file can be divided and processed by multiple Maps. After data is stored in the output directory, the number of saved files is the same as that of Maps. The file name format is **import_part**_*xxxx*, where *xxxx* is a unique random number generated by the system.	FILE
Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. This parameter cannot be set when **Extractor size** is set. The value must be less than or equal to 3000.	20
Extractor size	Size of data processed by Maps that are started in a MapReduce task of a data configuration operation. The unit is MB. The value must be greater than or equal to 100. The recommended value is 1000. This parameter cannot be set when **Extractors** is set. When a relational database connector is used, **Extractor Size** is unavailable. You need to set **Extractors**.	-

Configure data transformation.

Step 10 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-66**.

Table 20-66 Input and output parameters of the operator

Input Type	Output Type
CSV File Input	Table Output

In **input**, drag **CSV File Input** to the grid. In **output**, drag **Table Output** to the grid. Use an arrow to connect **CSV File Input** to **Table Output**.

Set data storage information and execute the job.

Step 11 Click **Next**. Configure the parameters on the **To** page displayed.

Table 20-67 Output parameters

Parameter	Description	Example Value
Schema name	Database schema name	dbo
Table name	Name of a database table that is used to save the final data of the transmission	test
Stage table name	Name of a temporary database table that is used to temporarily store data during transmission. The fields in the temporary table must be the same as those in the table specified by **Table Name**.	db_test
DB type	Type of the database. The options are **MOT** and other databases that can be connected through JDBC.	MOT

Parameter	Description	Example Value
MOT insert type	This parameter is available only when **Database Type** is set to **MOT**. Select an import mode based on service requirements. **NOTE** • Mode of importing data to the database. The options are **TOTAL**, **INCREMENT**, and **INSERT**. **TOTAL**: full import. The data version is **0** by default. The version of newly written data is **1**. When new data is imported to the database, data with the same primary key is updated, data with different primary keys is inserted, and all original data whose version is 0 is deleted. The version of the data that is newly written next time is 0, and the data versions are updated alternately in sequence. **INCREMENT**: incremental import. Data with the same primary key is updated, data with different primary keys is inserted, and the original data is retained. **INSERT**: common import. Data is inserted. If the primary key is duplicate, the task fails. • If this parameter is set to **TOTAL** or **INCREMENT**, ensure that there is the **CUR_VER_FLAG** field available in the service data table for version control. For example: **CREATE TABLE F_ACCOUNT1 (** **ORG_NBR smallint NOT NULL,** **ACT_NBR varchar NOT NULL,** **CLT_NBR varchar NOT NULL,** **BRF_NAM varchar,** **CUR_VER_FLAG tinyint DEFAULT '0' NOT NULL,** **CONSTRAINT IDX_F_ACCOUNT1_PKEY PRIMARY KEY (CLT_NBR,ORG_NBR)** **);**	TOTAL

Step 12 Click **Save and Run** to save and run the job.

View the job execution result.

Step 13 Go to the Loader web UI. When **Status** is **Succeeded**, the job is complete.

----End

20.5.7 Using Loader to Export Data from HBase to a Relational Database

Scenario

Use Loader to export data from HBase to a relational database.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the HBase tables or phoenix tables that are used during job execution.
- You have obtained the username and password of the relational database.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.
- Before the operation, perform the following steps:
  a. Obtain the JAR package of the relational database driver and save it to the following directory on the active and standby Loader nodes: **${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib**.
  b. Run the following command on the active and standby nodes as user root to modify the permission:

     cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

     chown omm:wheel *JAR package name*

     chmod 600 *JAR package name*
  c. Log in to FusionInsight Manager. Choose **Cluster** > **Service** > **Loader** > **More** > **Restart**. Enter the password of the administrator to restart the Loader service.

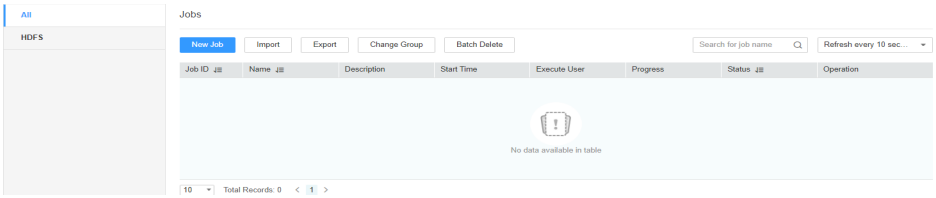
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-65 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-66 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Export**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **generic-jdbc-connector** or dedicated database connector (oracle-connector, oracle-partition-connector or mysql-fastpath-connector), set connection parameters, and click **Test** to verify whether the connection is available. When "**Test Success**" is displayed, click **OK**.

NOTE

- For connection to relational databases, general database connectors (generic-jdbc-connector) or dedicated database connectors (oracle-connector, oracle-partition-connector, and mysql-fastpath-connector) are available. However, compared with general database connectors, dedicated database connectors perform better in data import and export because they are optimized for specific database types.
- When **mysql-fastpath-connector** is used, the **mysqldump** and **mysqlimport** commands of MySQL must be available on NodeManagers, and the MySQL client version to which the two commands belong must be compatible with the MySQL server version. If the two commands are unavailable or the versions are incompatible, see **http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**. Install the MySQL client applications and tools.

Table 20-68 generic-jdbc-connector connection parameters

Parameter	Description	Example Value
Name	Specifies the name of a relational database connection.	dbName
JDBC Driver Class	Specifies the name of a Java database connectivity (JDBC) driver class.	oracle.jdbc.driver.OracleDriver
JDBC Connection String	Specifies the JDBC connection string.	jdbc:oracle:thin:@//10.16.0.1:1521/oradb
Username	Specifies the username for connecting to the database.	omm
Password	Specifies the password for connecting to the database.	xxxx
JDBC Connection Properties	JDBC connection attribute. Click **Add** to manually add the attribute. • Name: connection attribute name • Value: connection attribute value	• Name: socketTimeout • Value: 20

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set **Source type** to **HBASE**.

Table 20-69 Data source parameters

Parameter	Description	Example Value
HBase instance	Specifies the HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.	HBase
Quantity	Specifies the number of maps that are started at the same time in a MapReduce job of a data configuration operation. The value must be less than or equal to 3000.	20

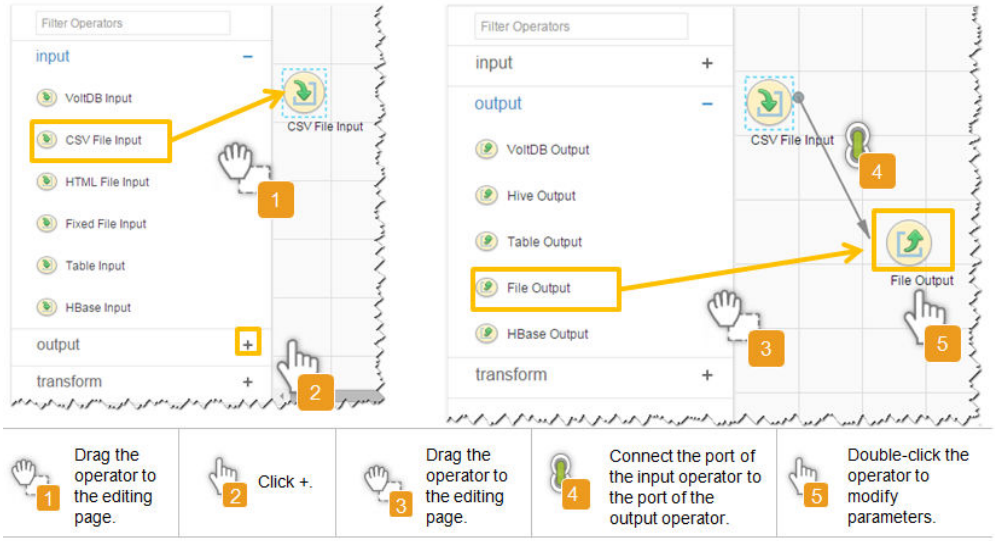
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-70**.

Table 20-70 Setting the input and output parameters of the operator

Input Type	Export Type
HBase input	Table output

Figure 20-67 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set the data storage mode.

Table 20-71 Parameter description

Parameter	Description	Example Value
Schema name	Specifies the database schema name.	dbo

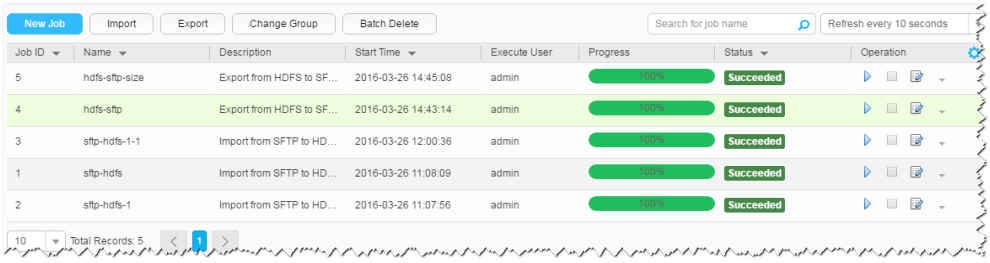
Parameter	Description	Example Value
Table name	Specifies the name of a database table that is used to save the final data of the transmission. **NOTE** Table names can be defined using macros. For details, see **Using Macro Definitions in Configuration Items** .	test
Temporary table	Specifies the name of a temporary database table that is used to save temporary data during the transmission. The fields in the table must be the same as those in the database specified by **Table name**. **NOTE** A temporary table is used to prevent dirty data from being generated in the destination table when data is exported to the database. Data is migrated from the temporary table to the destination table only after all data is successfully written to the temporary table. Using temporary tables increases the job execution time.	tmp_test

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the **Loader WebUI**. When **Status** is **Succeeded**, the job is complete.

Figure 20-68 Viewing a job



----End

20.5.8 Using Loader to Export Data from Hive to a Relational Database

Scenario

Use Loader to export data from Hive to a relational database.

Prerequisites

- You have obtained the service username and password for creating a Loader job.
- You have had the permission to access the Hive tables that are used during job execution.
- You have obtained the username and password of the relational database.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.
- Before the operation, perform the following steps:

  a. Obtain the JAR package of the relational database driver and save it to the following directory on the active and standby Loader nodes: **${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib**.

  b. Run the following command on the active and standby nodes as user root to modify the permission:

  cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

  chown omm:wheel *JAR package name*

  chmod 600 *JAR package name*

  c. Log in to FusionInsight Manager. Choose **Cluster** > **Service** > **Loader** > **More** > **Restart**. Enter the password of the administrator to restart the Loader service.

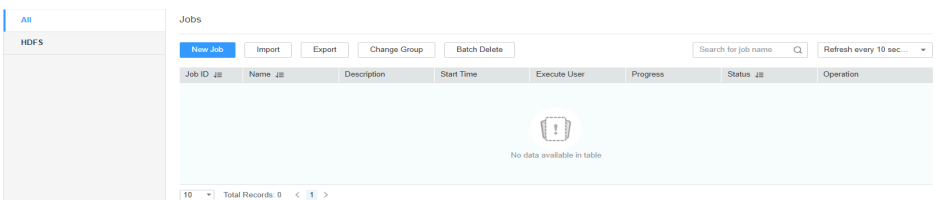
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(*Node name*, Active)**. The Loader web UI is displayed.

Figure 20-69 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-70 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Export**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **generic-jdbc-connector** or dedicated database connector (oracle-connector, oracle-partition-connector or mysql-fastpath-connector), set connection parameters, and click **Test** to verify whether the connection is available. When "**Test Success**" is displayed, click **OK**.

NOTE

- For connection to relational databases, general database connectors (generic-jdbc-connector) or dedicated database connectors (oracle-connector, oracle-partition-connector, and mysql-fastpath-connector) are available. However, compared with general database connectors, dedicated database connectors perform better in data import and export because they are optimized for specific database types.
- When **mysql-fastpath-connector** is used, the **mysqldump** and **mysqlimport** commands of MySQL must be available on NodeManagers, and the MySQL client version to which the two commands belong must be compatible with the MySQL server version. If the two commands are unavailable or the versions are incompatible, see **http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**. Install the MySQL client applications and tools.

Table 20-72 generic-jdbc-connector connection parameters

Parameter	Description	Example Value
Name	Specifies the name of a relational database connection.	dbName

Parameter	Description	Example Value
JDBC Driver Class	Specifies the name of a Java database connectivity (JDBC) driver class.	oracle.jdbc.driver.OracleDriver
JDBC Connection String	Specifies the JDBC connection string.	jdbc:oracle:thin:@//10.16.0.1:1521/oradb
Username	Specifies the username for connecting to the database.	omm
Password	Specifies the password for connecting to the database.	xxxx
JDBC Connection Properties	JDBC connection attribute. Click **Add** to manually add the attribute. • Name: connection attribute name • Value: connection attribute value	• Name: socketTimeout • Value: 20

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set **Source type** to **HIVE**.

Table 20-73 Data source parameters

Parameter	Description	Example Value
Hive instance	Specifies the Hive service instance that Loader selects from all available Hive service instances in the cluster. If the selected Hive service instance is not added to the cluster, the Hive job cannot run properly.	hive
Quantity	Specifies the number of maps that are started at the same time in a MapReduce job of a data configuration operation. The value must be less than or equal to 3000.	20

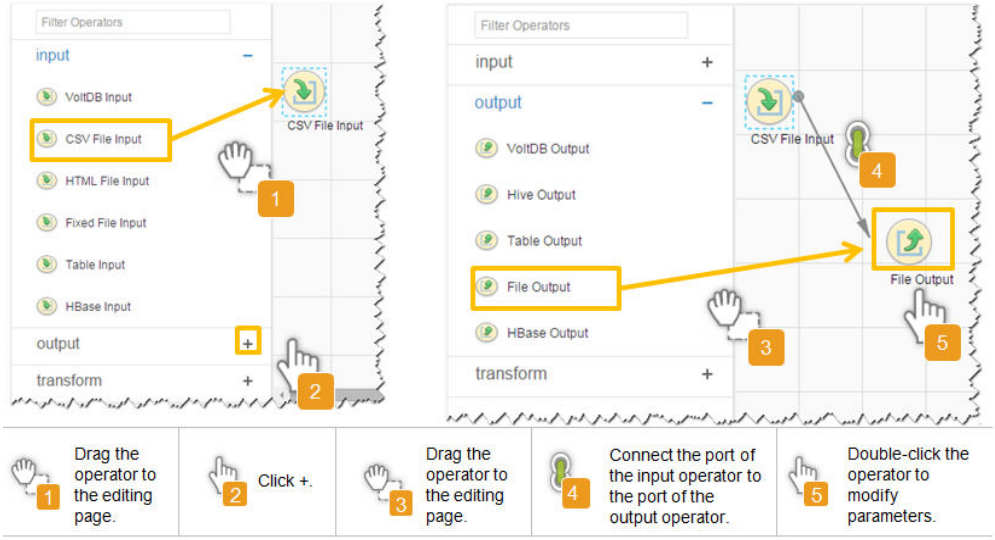
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-74**.

Table 20-74 Setting the input and output parameters of the operator

Input Type	Export Type
Hive input	Table output

Figure 20-71 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set the data storage mode.

Table 20-75 Parameter description

Parameter	Description	Example Value
Schema name	Specifies the database schema name.	dbo
Table name	Specifies the name of a database table that is used to save the final data of the transmission. **NOTE** Table names can be defined using macros. For details, see **Using Macro Definitions in Configuration Items** .	test

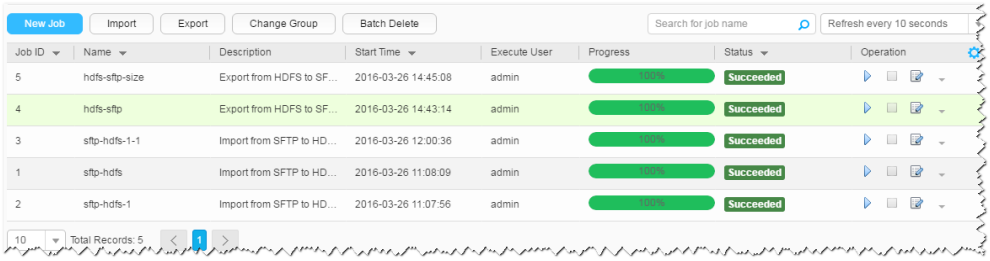
Parameter	Description	Example Value
Temporary table	Specifies the name of a temporary database table that is used to save temporary data during the transmission. The fields in the table must be the same as those in the database specified by **Table name**. **NOTE** A temporary table is used to prevent dirty data from being generated in the destination table when data is exported to the database. Data is migrated from the temporary table to the destination table only after all data is successfully written to the temporary table. Using temporary tables increases the job execution time.	tmp_test

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the **Loader WebUI**. When **Status** is **Succeeded**, the job is complete.

Figure 20-72 Viewing a job



----End

20.5.9 Using Loader to Export Data from HBase to HDFS or OBS

Scenario

This section describes how to use Loader to export data from HBase to HDFS or OBS.

Prerequisites

- You have obtained the service user name and password for creating a Loader job.
- You have had the permission to access the HDFS or OBS directories and data involved in job execution.

- You have had the permission to access the HBase tables or phoenix tables that are used during job execution.
- No disk space alarm is reported, and the available disk space is sufficient for importing and exporting data.
- If a configured task requires the Yarn queue function, the user must be authorized with related Yarn queue permission.
- The user who configures a task must obtain execution permission on the task and obtain usage permission on the related connection of the task.

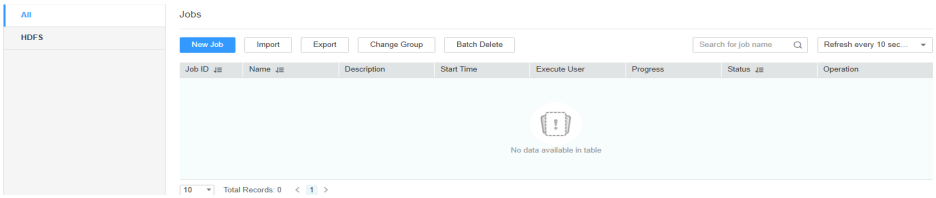
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(*Node name*, Active)**. The Loader web UI is displayed.

Figure 20-73 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 20-74 Basic Information

1. Set **Name** to the name of the job.
2. Set **Type** to **Export**.
3. Set **Group** to the group to which the job belongs. No group is created by default. You need to click **Add** to create a group and click **OK** to save the created group.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 3 In the **Connection** area, click **Add** to create a connection, set **Connector** to **hdfs-connector**, set connection parameters, and click **Test** to verify whether the connection is available. When "**Test Success**" is displayed, click **OK**.

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set **Source type** to **HBASE**.

Table 20-76 Parameter description

Parameter	Description	Example
HBase instance	Specifies the HBase service instance that Loader selects from all available HBase service instances in the cluster. If the selected HBase service instance is not added to the cluster, the HBase job cannot be run properly.	HBase
Quantity	Specifies the number of maps that are started at the same time in a MapReduce job of a data configuration operation. The value must be less than or equal to 3000.	20

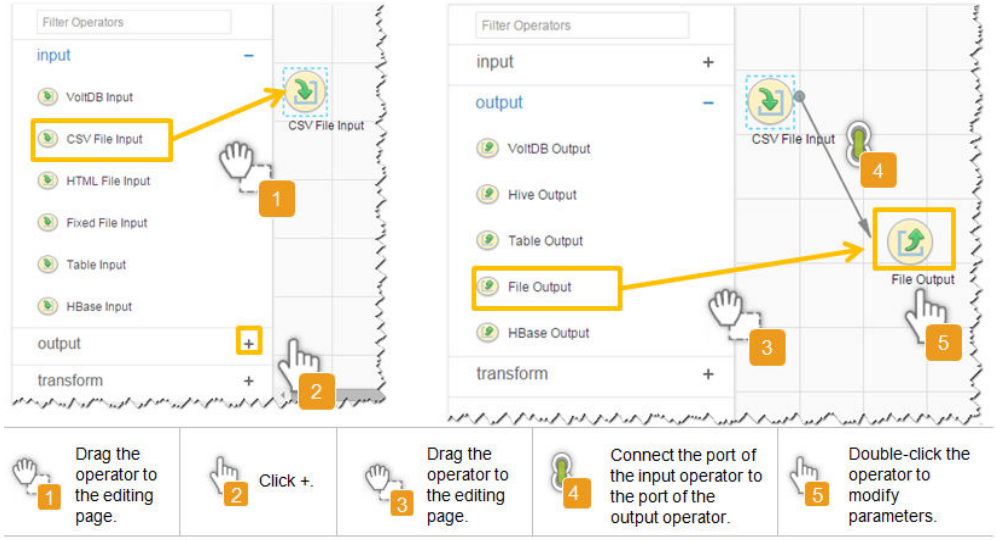
Setting Data Transformation

Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-77**.

Table 20-77 Setting the input and output parameters of the operator

Input Type	Export Type
HBase input	File output

Figure 20-75 Operator operation procedure



Setting Data Storage Information and Executing the Job

Step 6 Click **Next**. On the displayed **To** page, set the data storage mode.

Table 20-78 Parameter description

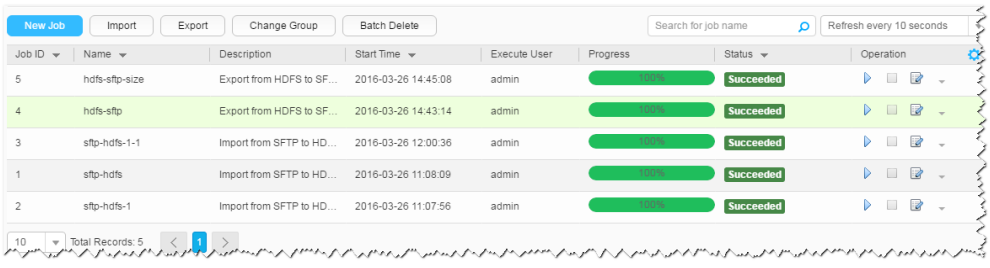
Parameter	Description	Example
Output path	Specifies the output directory or file name of the export file in the HDFS or OBS. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/user/test
File Format	Specifies the file export type. • **TEXT_FILE**: imports a text file and stores it as a text file. • **SEQUENCE_FILE**: imports a text file and stores it as a **sequence file**. • **BINARY_FILE**: imports files of any format by using binary streams.	TEXT_FILE
Compression codec	Specifies the compression format of files exported to HDFS or OBS. Select a format from the drop-down list. If you select **NONE** or do not set this parameter, data is not compressed.	NONE

Step 7 Click **Save and run** to save and run the job.

Checking the Job Execution Result

Step 8 Go to the **Loader WebUI**. When **Status** is **Succeeded**, the job is complete.

Figure 20-76 Viewing a job



----End

20.5.10 Using Loader to Export Data from HDFS to ClickHouse

This section applies to MRS 3.3.0 or later.

Scenario

Use Loader to export data from HDFS to ClickHouse.

Prerequisites

- A role has been created on FusionInsight Manager and granted the management permission on ClickHouse logical clusters and Loader job grouping permission. A service user for Loader jobs has been created, associated with the role, and added the user group **yarnviewgroup**.
- A replicated table and a distributed table have been created by referring to **ClickHouse Client Practices** and a user has been assigned the permission to perform operations on the tables during job execution. The replicated table has been selected when data is exported.
- No ClickHouse alarm is generated.

Procedure

Make preparations.

Step 1 Obtain the **clickhouse-jdbc-*.jar** file from the ClickHouse installation directory and save it to **${BIGDATA_HOME}/FusionInsight_Porter_*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib** on the active and standby Loader nodes.

Step 2 Run the following command on the active and standby nodes as user **root** to modify the permission:

cd ${BIGDATA_HOME}/FusionInsight_Porter_*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

chown omm:wheel *JAR file name*

chmod 600 *JAR file name*

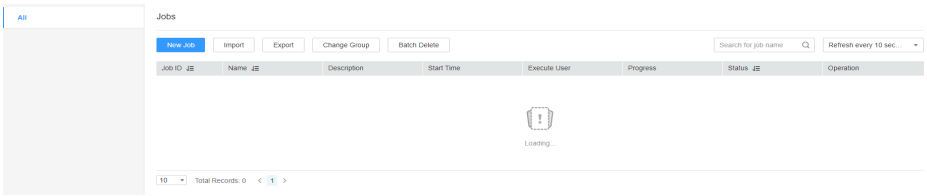
Step 3 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Loader**. On the **Dashboard** tab page that is displayed, click **More** and select **Restart Service**. In the displayed dialog box, enter the administrator password to restart the Loader service.

Configure basic job information.

Step 4 Access the Loader web UI.

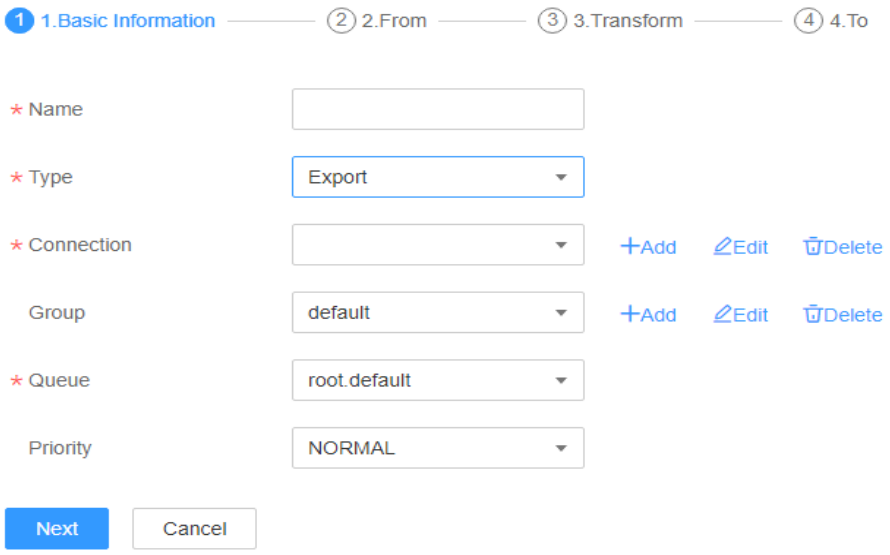
1. Log in to FusionInsight Manager.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-77 Loader web UI



Step 5 Click **New Job**. Configure basic job parameters on the **Basic Information** page displayed.

Figure 20-78 Basic Information page



1. Enter a job name in **Name**.
2. Set **Type** to **Export**.
3. Set **Group** to the group to which the job belongs. There is no group created by default. Click **Add**, enter the group name, and click **OK**.
4. Set **Queue** to the Yarn queue that executes the job. The default value is **root.default**.
5. Set **Priority** to the priority of the Yarn queue that executes the job. The default value is **NORMAL**. The options are **VERY_LOW**, **LOW**, **NORMAL**, **HIGH**, and **VERY_HIGH**.

Step 6 In the **Connection** area, click **Add** to create a connection, set **Connector** to **clickhouse-connector**, configure connection parameters, and click **Test** to verify whether the connection is available. When "Test Success" is displayed, click **OK**. For details about parameter settings, see **Table 20-79**.

Table 20-79 clickhouse-connector connection parameters

Parameter	Description	Example Value
Name	Name of a relational database connection	clickhouse_jdbc_test

Parameter	Description	Example Value
ClickHouse Connection String	• Kerberos authentication has been enabled for the cluster. The JDBC connection string format is **jdbc:clickhouse://** *Database IP address*:*Database port number*/*Database name*?**ssl=true&sslmode=none**. • Kerberos authentication is disabled for the cluster. The JDBC connection string format is **jdbc:clickhouse://** *Database IP address*:*Database port number*/*Database name*.	• Kerberos authentication has been enabled for the cluster: **jdbc:clickhouse://10.10.10.10:21426/test?ssl=true&sslmode=none** • Kerberos authentication is disabled for the cluster: **jdbc:clickhouse://10.10.10.10:21423/test?ssl=false&sslmode=none**

Parameter	Description	Example Value
	NOTE • *Database IP address*: To obtain the IP address of the ClickHouseBalancer instance, log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse**, and click **Instance**. • Database port number: – To obtain the port number of a cluster with Kerberos authentication enabled, log in to FusionInsight Manager, choose **Cluster** > **Services**, click **Logical Cluster**, view the logical cluster, and obtain the value of **Ssl Port** in **HTTP Balancer Port**. – To obtain the port number of a cluster with Kerberos authentication disabled, log in to FusionInsight Manager, choose **Cluster** > **Services**, click **Logical Cluster**, view the logical cluster, and obtain the value of **Port** in **HTTP Balancer Port**.	
Username	Username for connecting to the database	root
Password	Password for connecting to the database	xxxx

Configure data source information.

Step 7 Click **Next**. On the **From** page displayed, set **Source type** to **HDFS**.

Table 20-80 Input parameters

Parameter	Description	Example Value
Input directory	Input path when data is exported from HDFS **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/user/test
Path filter	Wildcard for filtering the directories in the input paths of the source files. **Input directory** is not used for filtering. Use commas (,) to separate multiple filter conditions. If this parameter is left blank, directories are not filtered. Regular expression filtering is not supported. • **?** matches a single character. • ***** indicates multiple characters. • Adding ^ before the condition indicates negated filtering, that is, file filtering.	*
File filter	Wildcard for filtering the file names of the source files. Use commas (,) to separate multiple filter conditions. This parameter cannot be left blank. Regular expression filtering is not supported. • **?** matches a single character. • ***** indicates multiple characters. • Adding ^ before the condition indicates negated filtering, that is, file filtering.	*
File type	File import type. The options are as follows: • **TEXT_FILE**: imports a text file and saves it as a text file. • **SEQUENCE_FILE**: imports a text file and saves it as a sequence file. • **BINARY_FILE**: imports files of any format using binary streams but not to process the files. **NOTE** When the file import type is set to **TEXT_FILE** or **SEQUENCE_FILE**, Loader automatically selects a decompression method based on the file name extension to decompress a file.	TEXT_FILE

Parameter	Description	Example Value
File split type	Whether to split source files by file name or size. The files obtained after the splitting are used as the input files of each Map in the MapReduce task for data export. • **FILE**: indicates that the source file is split by file. That is, each Map processes one or multiple complete files, the same source file cannot be allocated to different Maps, and the source file directory structure is retained after data import. • **SIZE**: indicates that the source file is split by size. That is, each Map processes input files of a certain size, and a source file can be divided and processed by multiple Maps. After data is stored in the output directory, the number of saved files is the same as that of Maps. The file name format is **import_part**_*xxxx*, where *xxxx* is a unique random number generated by the system.	FILE
Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. This parameter cannot be set when **Extractor size** is set. The value must be less than or equal to 3000.	20
Extractor size	Size of data processed by Maps that are started in a MapReduce task of a data configuration operation. The unit is MB. The value must be greater than or equal to 100. The recommended value is 1000. This parameter cannot be set when **Extractors** is set. When a relational database connector is used, **Extractor Size** is unavailable. You need to set **Extractors**.	-

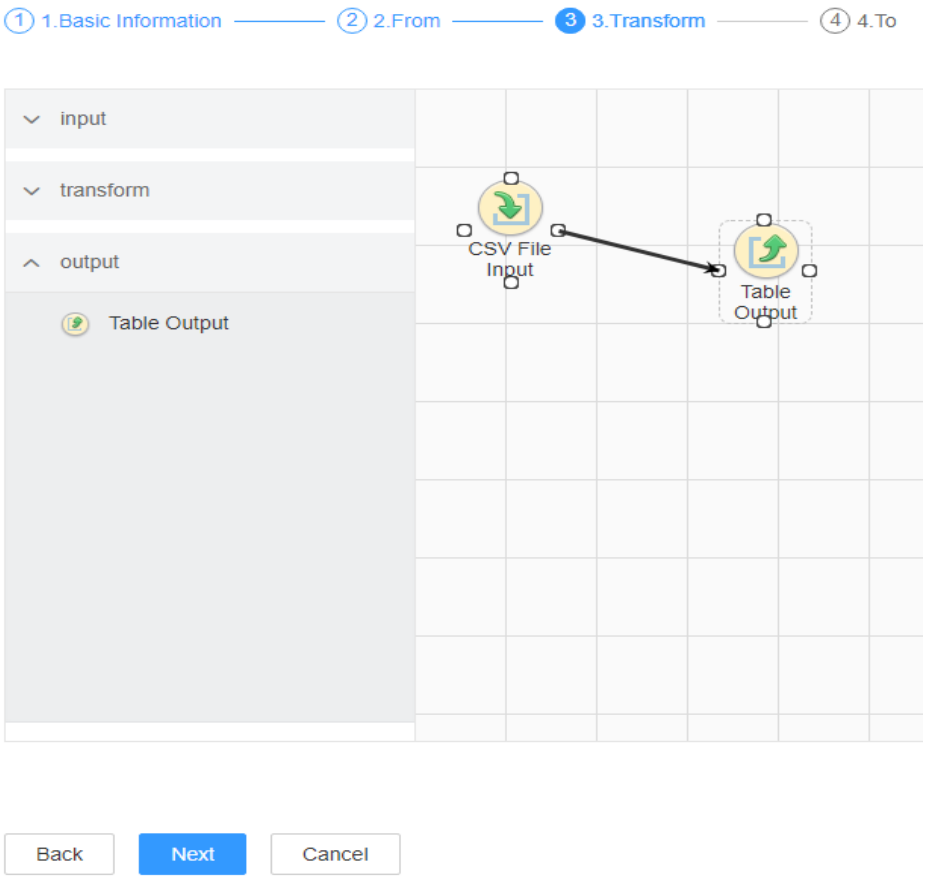
Configure data transformation.

Step 8 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Loader Operator Help** and **Table 20-81**.

Table 20-81 Input and output parameters of the operator

Input Type	Output Type
CSV File Input	Table Output

Figure 20-79 Operator selection



Set data storage information and execute the job.

Step 9 Click **Next**. On the displayed **To** page, set the data storage mode.

Table 20-82 Output parameters

Parameter	Description	Example Value
Table Name	Name of a database table that is used to save the final data of the transmission **NOTE** You can use macros to define table names. For details, see **Using Macro Definitions in Configuration Items** .	test

Step 10 Click **Save and Run** to save and run the job.

View the job execution result.

Step 11 Go to the Loader web UI. When **Status** is **Succeeded**, the job is complete.

Figure 20-80 Viewing a job

Step 12 On the ClickHouse client, check whether the data in the ClickHouse table is the same as that in HDFS.

----End

20.6 Managing Loader Jobs

20.6.1 Migrating Loader Jobs in Batches

Scenario

Loader allows jobs to be migrated in batches from a group (source group) to another group (target group).

Prerequisites

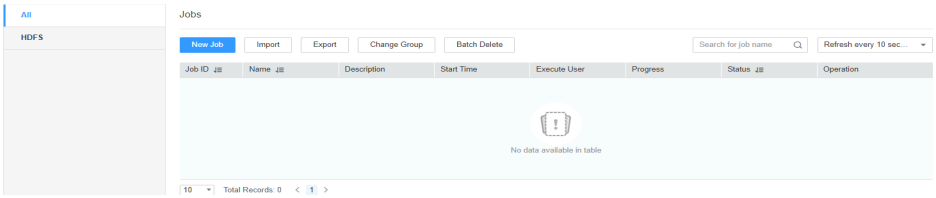
- The source group and target group exist.
- The current user has the **Group Edit** permission for the source group and target group.
- The current user has the **Jobs Edit** permission for the source group or the **Edit** permission for the jobs to be migrated.

Procedure

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-81 Loader web UI



Step 2 Click **Change Group**. The Job Migration page is displayed.

Step 3 In **Source group**, select the group to which the jobs to be migrated belong; in **target group**, select the group to which the jobs are to be migrated.

Step 4 Set **Select Change Type** to a migration type.

- **All**: migrates all the jobs in the source group to the target group.
- **Specify Job**: migrates the specified jobs in the source group to the target group. Select **Specify Job**. In the job list, select the jobs to be migrated.

Step 5 Click **OK** to start job migration. In the displayed dialog box, if the progress bar is 100%, the job migration is complete.

----End

20.6.2 Deleting Loader Jobs in Batches

Scenario

Loader allows existing jobs to be deleted in batches.

Prerequisites

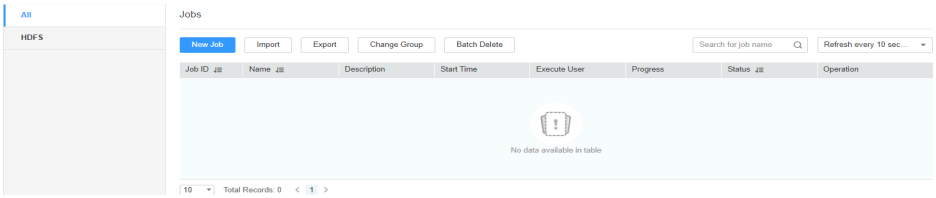
The current user has the **Edit** permission for the jobs to be deleted or the **Jobs Edit** permission for the group to which the jobs belong.

Procedure

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-82 Loader web UI



Step 2 Click **Batch Delete**. The Batch Delete page is displayed.

Step 3 Set **Batch Delete** to a job deletion type.

- **ALL**: deletes all jobs.
- **Specify Job**: deletes specified jobs. Select **Specify Job**. In the job list, select the jobs to be deleted.

Step 4 Click **OK** to start the job deletion. In the displayed dialog box, if the progress bar is 100%, the job deletion is complete.

----End

20.6.3 Importing Loader Jobs in Batches

Scenario

Loader allows all jobs of a configuration file to be imported in batches.

Prerequisites

The current user has the **Jobs Edit** permission of the group to which the jobs to be imported belong.

NOTE

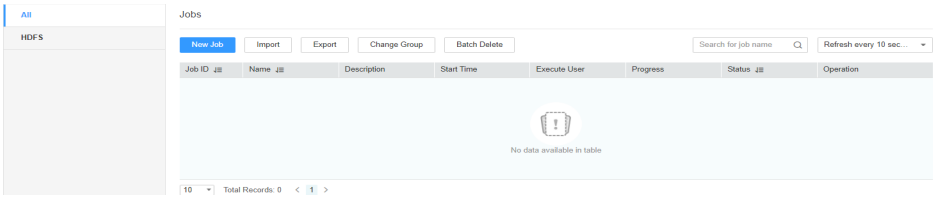
If the group to which the jobs to be imported belong does not exist, the group is automatically created first. The current user is the creator of the group and has the **Jobs Edit** permission of the group.

Procedure

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-83 Loader web UI



Step 2 Click **Import**. The Export Job page is displayed.

Step 3 On the **Import** page, specify the path of the configuration file whose jobs are to be imported.

Step 4 Click **Upload** to start the job import. In the displayed dialog box, if the progress bar is 100%, the job import is complete.

----End

20.6.4 Exporting Loader Jobs in Batches

Scenario

Loader allows existing jobs to be exported in batches.

Prerequisites

The current user has the **Edit** permission for the jobs to be exported or the **Jobs Edit** permission of the group to which the jobs belong.

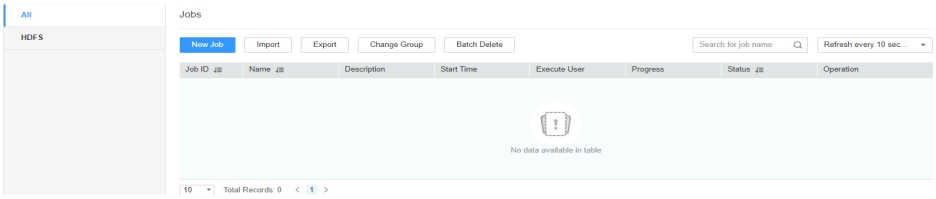
Procedure

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.

2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-84 Loader web UI



Step 2 Click **Export**. The job export page is displayed.

Step 3 Set **Batch Delete** to a job export type.

- **ALL**: exports all jobs.
- **Specify Job**: exports specified jobs. Select **Specify Job**. In the job list, select the jobs to be exported.
- **Specify Group**: exports all the jobs in a specified group. Select **Specify Group**. In the group list, select the group whose jobs are to be exported.

Export Password: exports the connector password. If this parameter is selected, the password is exported as an encrypted string.

Step 4 Click **OK** to start the job export. In the displayed dialog box, if the progress bar is 100%, the job import is complete.

----End

20.6.5 Viewing Historical Information About a Loader Job

Scenario

Query the execution status and execution duration of a Loader job during routine maintenance. You can perform the following operations on the job:

- Dirty Data: Query data that fails to be processed or data that is filtered out during job execution, and check which source data does not meet transformation or cleaning rules.
- Logs: Query log information about job execution in MapReduce.

Prerequisites

You have obtained the username and password for logging in to the Loader WebUI.

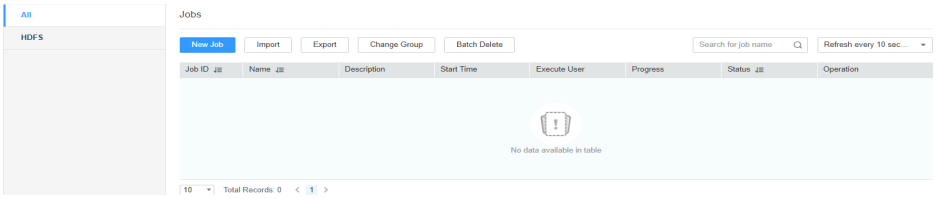
Procedure

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.

3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-85 Loader web UI



Step 2 Query historical records of a Loader job.

1. Locate the row that contains the job to be viewed.
2. Click **More** and select **View History** to view the job execution history.

Figure 20-86 Viewing historical records

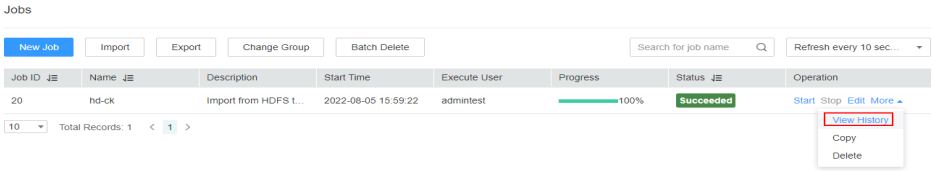


Table 20-83 Parameters

Name	Description
Rows/Files Read	Indicates the number of rows (files) read from the input source.
Rows/Files Written	Number of rows (files) written to the output source.
Rows/Files Skipped	– Indicates the number of bad rows (files) recorded during transformation. The input format is incorrect, so transformation cannot be performed. – Number of rows that are skipped after filtering conditions are configured during conversion.

----End

20.6.6 Purging Historical Loader Data

This section applies to MRS 3.2.0 or later.

Scenario

Loader accumulates a large amount of historical data during service running. The historical data may affect job submission, running, and status query, and even

cause page freezing and job running failures. Therefore, you need to properly configure the historical data pruge policy based on the Loader service data volume.

Procedure

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **Loader** and click the **Configurations** tab and then **All Configurations**. In the navigation pane on the left, choose **LoaderServer(Role)** > **Purge**. Then adjust the parameter settings shown in the following figure by referring to **Table 1**.

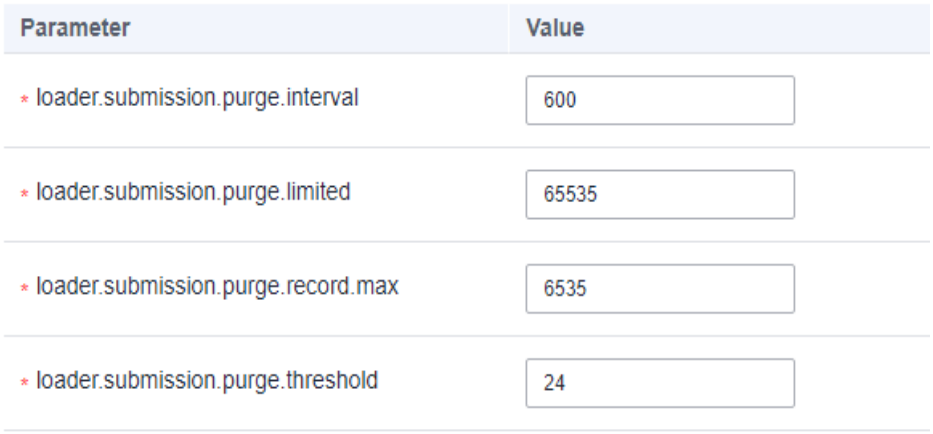


Table 20-84 Parameters for purging historical Loader data

Parameter	Description	Recommended Value
loader.submission.purge.interval	Interval for invoking the purge task, in minutes.	60
loader.submission.purge.limited	Number of submissions that are retained during the purge. This prevents historical job records from being totally purged.	0
loader.submission.purge.record.max	Maximum number of records that can be retained in a Loader job. Value **0** indicates that the number is not limited.	7
loader.submission.purge.threshold	Duration for retaining historical records, in hours.	24

Step 3 Click **Save**.

Step 4 Click **Dashboard** to go to the Loader service page. Click **More** and select **Restart Service**. Verify the identity and click **OK**. Wait until the restart is successful.

----End

20.6.7 Managing Loader Links

Scenario

You can create, view, edit, and delete links on the Loader page.

Creating a Connection

Step 1 Log in to the service page.

Log in to FusionInsight Manager (for details, see **Accessing FusionInsight Manager**) and choose **Cluster** > **Services**.

Step 2 Select **Loader**. On the right of **Loader WebUI**, click the link to open the Loader web UI.

Step 3 On the Loader page, click **New job**.

Step 4 Click **Add** next to **Connection** and set connection parameters.

For details about the parameters, see **Loader Connection Configuration**.

Step 5 Click **OK**.

If connection configurations, for example, IP address, port, and access user information, are incorrect, the connection will fail to be verified and saved.

NOTE

You can click **Test** to immediately check whether the connection is available.

----End

Viewing a Connection

Step 1 On the Loader page, click **New job**.

Step 2 Click the drop-down list of **Connection** to view the connections you have created.

----End

Editing a Connection

Step 1 On the Loader page, click **New job**.

Step 2 Select the name of the connection to be edited from the **Connection** drop-down list.

Step 3 Click **Edit** next to **Connection**.

Step 4 On the dialog box displayed, modify the connection parameters based on service requirements.

Step 5 Click **Test**.

- If the test is successful, go to **Step 6**.
- If the test fails, repeat **Step 4**.

Step 6 Click **Save**.

If a Loader job has integrated into a Loader link, editing the link parameters may affect Loader running.

----End

Deleting a Connection

Step 1 On the Loader page, click **New job**.

Step 2 Select the name of the connection to be deleted from the **Connection** drop-down list.

Step 3 Click **Delete**.

Step 4 In the displayed dialog box, click **OK**.

If a Loader job has integrated a Loader connection, the connection cannot be deleted.

----End

Loader Connection Configuration

Loader supports the following connections:

- **generic-jdbc-connector**: For details about parameter settings, see **Table 20-85**.
- **ftp-connector**: For details about parameter settings, see **Table 20-86**.
- **sftp-connector**: For details about parameter settings, see **Table 20-87**.
- **hdfs-connector**: For details about parameter settings, see **Table 20-88**.
- **oracle-connector**: For details about parameter settings, see **Table 20-89**.
- **mysql-fastpath-connector**: For details about parameter settings, see **Table 20-91**.
- **oracle-partition-connector**: For details about parameter settings, see **Table 20-90**.

Table 20-85 generic-jdbc-connector configuration

Parameter	Description
Name	Name of a Loader connection
Connector	Select **generic-jdbc-connector**.

Parameter	Description
JDBC Driver Class	JDBC driver classes are as follows: • oracle: **oracle.jdbc.driver.OracleDriver** • SQLServer: **com.microsoft.jdbc.sqlserver.SQLServerDriver** • mysql: **com.mysql.jdbc.Driver** • postgresql: **org.postgresql.Driver** • gaussdb200: **com.huawei.gauss200.jdbc.Driver**
JDBC Connection String	Database access address, which can be an IP address or domain name. Enter the database connection string. In the following examples, **10.10.10.10** indicates the IP address and **test** indicates the database name. • oracle: **jdbc:oracle:thin:@10.10.10.10:1521:orcl** • SQLServer: **jdbc:microsoft:sqlserver://10.10.10.10:1433;DatabaseName=test** • mysql: **jdbc:mysql://10.10.10.10/test?&useUnicode=true&characterEncoding=GBK** • postgresql: **jdbc:postgresql://10.10.10.10:5432/test** • gaussdb200: **jdbc:gaussdb://10.10.10.10:15400/test** (**15400** is an example port number.)
Username	Username for accessing the database
Password	Password of the user. Use the actual password.

Table 20-86 ftp-connector configuration

Parameter	Description
Name	Name of a Loader connection
Connector	Select **ftp-connector**.
FTP Mode	Select **ACTIVE** or **PASSIVE**.
FTP Protocol	Select: • FTP • SSL_EXPLICIT • SSL_IMPLICIT • TLS_EXPLICIT • TLS_IMPLICIT
File Name Encoding Type	Encoding type of the file name or file path.

Table 20-87 sftp-connector configuration

Parameter	Description
Name	Name of a Loader connection
Connector	Select **sftp-connector**.

Table 20-88 hdfs-connector configuration

Parameter	Description
Name	Name of a Loader connection
Connector	Select **hdfs-connector**.

Table 20-89 oracle-connector configuration

Parameter	Description
Name	Name of a Loader connection
Connector	Select **oracle-connector**.
JDBC Connection String	Enter the connection string for connecting to the database, for example, jdbc:oracle:thin:@IP:port:database.
Username	Username for accessing the database
Password	Password of the user. Use the actual password.

Table 20-90 oracle-partition-connector configuration

Parameter	Description
Name	Name of a Loader connection
Connector	Select **oracle-partition-connector**.
JDBC Driver Class	Enter **com.microsoft.jdbc.sqlserver.SQLServerDriver**.
JDBC Connection String	Enter the connection string for connecting to the database, for example, jjdbc:oracle:thin:@*IP*:*port*:*database*.
Username	Username for accessing the database
Password	Password of the user. Use the actual password.

Table 20-91 mysql-fastpath-connector configuration

Parameter	Description
Name	Name of a Loader connection
Connector	Select **mysql-fastpath-connector**. **NOTICE** When **mysql-fastpath-connector** is used, the **mysqldump** and **mysqlimport** commands of MySQL must be available on NodeManagers, and the MySQL client version to which the two commands belong must be compatible with the MySQL server version. If the two commands are unavailable or the versions are incompatible, see **http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**. Install the MySQL client applications and tools. For example, you need to install the following RPM packages in the RHEL-x86 system (select the package version based on the site requirements): • mysql-community-client-5.7.23-1.el7.x86_64.rpm • mysql-community-common-5.7.23-1.el7.x86_64.rpm • mysql-community-devel-5.7.23-1.el7.x86_64.rpm • mysql-community-embedded-5.7.23-1.el7.x86_64.rpm • mysql-community-libs-5.7.23-1.el7.x86_64.rpm • mysql-community-libs-compat-5.7.23-1.el7.x86_64.rpm
JDBC Connection String	Enter the connection string for connecting to the database, for example, jdbc:mysql://IP/database?.&useUnicode=true&characterEncoding=GBK."
Username	Username for accessing the database
Password	Password of the user. Use the actual password.

20.7 Loader O&M Management

20.7.1 Loader Common Configuration Parameters

Navigation Path

For details about the how to set parameters, see **Modifying Cluster Service Configuration Parameters**.

Parameter Description

Table 20-92 Common Loader parameters

Parameter	Description	Default Value	Value Range
mapreduce.client.submit.file.replication	Number of copies of the job files that the MapReduce task depends on in HDFS. If the number of DataNodes in the cluster is less than the value of this parameter, the number of copies is equal to the number of DataNodes. If the number of DataNodes is greater than or equal to the value of this parameter, the number of copies is the value of this parameter.	10	3 to 256
loader.fault.tolerance.rate	Error tolerance. If the value is greater than 0, the error tolerance mechanism is enabled. When enabling the fault tolerance mechanism, you are advised to set the number of Map jobs to be greater than or equal to 3. It is recommended that this function be used when the job data volume is large.	0	0 to 1.0
loader.input.field.separator	Default input field separator. The parameter value takes effect only when input and output conversion steps are configured. The conversion steps can be left blank. If no separators are configured in job conversion steps, the default separator is used.	,	-

Parameter	Description	Default Value	Value Range
loader.input.line.separator	Default input line separator. The parameter value takes effect only when input and output conversion steps are configured. The conversion steps can be left blank. If no separators are configured in job conversion steps, the default separator is used.	-	-
loader.output.field.separator	Default output field separator. The parameter value takes effect only when input and output conversion steps are configured. The conversion steps can be left blank. If no separators are configured in job conversion steps, the default separator is used.	,	-
loader.output.line.separator	Line separator of data that Loader outputs	-	-

NOTE

- Because it needs time to calculate the fault tolerance rate, you are recommended to use the **loader.fault.tolerance.rate** parameter when the job runtime is longer than 2 minutes to ensure user experience.
- Default separators are configured for the parameters in the preceding table for Loader. If separators are configured in the conversion steps for the jobs, the separators in the conversion steps will be used. If separators are not configured in the conversion steps, the default separators will be used.

20.7.2 Loader Log Overview

Log Description

Log path: The default storage path of Loader log files is **/var/log/Bigdata/loader/Log category**.

- runlog: /var/log/Bigdata/loader/runlog (run logs)
- scriptlog: /var/log/Bigdata/loader/scriptlog/ (script execution logs)
- catalina: /var/log/Bigdata/loader/catalina (Tomcat startup and stop logs)
- audit: /var/log/Bigdata/loader/audit (audit logs)

Log archive rule:

The automatic compression and archiving function are enabled for Loader run logs and audit logs. By default, when the size of a log file exceeds 10 MB, the log

file is automatically compressed into a log file named in the following rule: *<Original log file name>-<yyyy-mm-dd_hh-mm-ss>.[ID]*.**log.zip**. A maximum of 20 latest compressed files are reserved. The number of compressed files can be configured on the Manager portal.

Table 20-93 Loader log list

Log Type	Log File Name	Description
Run log	loader.log	Loader system log file that records most of the logs generated when the TelcoFS system is running.
	loader-omm-***-pid***-gc.log.*.current	Loader process GC log file
	sqoopInstanceCheck.log	Loader instance health check log file
Audit log	default.audit	Loader operation audit log file that records operations such as adding, deleting, modifying, and querying jobs and user login
Tomcat log	catalina.out	Tomcat run log file.
	catalina. <yyyy-mm-dd >.log	Tomcat run log file
	host-manager. <yyyy-mm-dd >.log	Tomcat run log file
	localhost_access_log. <yyyy-mm-dd >.txt	Tomcat run log file
	manager <yyyy-mm-dd >.log	Tomcat run log file
	localhost. <yyyy-mm-dd >.log	Tomcat run log file
Script log	postInstall.log	Loader installation script log file Log file generated during the execution of the Loader installation script (**postInstall.sh**)

Log Type	Log File Name	Description
	preStart.log	Pre-startup script log file of the Loader service During startup of the Loader service, a series of preparation operations are first performed (by executing **preStart.sh**), such as generating the keytab file. This log file records information about these operations
	loader_ctl.log	Log file generated when Loader executes the service start and stop script (**sqoop.sh**)

Log Level

Table 20-94 describes the log levels provided by Loader. The priorities of log levels are ERROR, WARN, INFO, and DEBUG in descending order. Logs whose levels are higher than or equal to the specified level are printed. The number of printed logs decreases as the specified log level increases.

Table 20-94 Log levels

Level	Description
ERROR	Error information about the current event processing.
WARN	Exception information about the current event processing.
INFO	Normal running status information about the system and events.
DEBUG	System information and system debugging information.

To modify log levels, perform the following operations:

Step 1 Go to the **All Configurations** page of Loader by referring to **Modifying Cluster Service Configuration Parameters**.

Step 2 On the menu bar on the left, select the log menu of the target role.

Step 3 Select a desired log level.

Step 4 Save the configuration. In the dialog box that is displayed, click **OK**. Then restart the service for the configuration to take effect.

----End

Log Formats

The following table lists the Loader log formats.

Table 20-95 Log formats

Log Type	Format	Example
Run log	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<Thread that generates the log>|<Message in the log>|<Location of the log event>	2015-06-29 14:54:35,553 | INFO | [localhost-startStop-1] | ConnectionRequestHandler initialized | org.apache.sqoop.handler.ConnectionRequestHandler.<init>(ConnectionRequestHandler.java:100)
Audit log	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|default|<Message in the log>|<Location of the log event>	2015-06-29 15:35:40,969 INFO default: UserName=admin, UserIP=10.52.0.111, Time=2015-06-29 15:35:40,969, Operation=submit, Resource=submission@21, Result=Failure, Detail={[reason:GET_SFTP_SESSION_FAILED:Failed to get sftp session - 10.162.0.35 (caused by: Auth cancel)]; [config:null]}

20.8 Loader Operator Help

20.8.1 Loader Operator Description

The "Operator Help" section applies to MRS 3.x and later versions.

Conversion Process

Loader reads data at the source end, uses an input operator to convert data into fields by certain rules, use a conversion operator to clean or convert the fields, and finally use an output operator to process the fields and export the output result to the target end.

- A job for performing data conversion can have only one input operator and one output operator.
- Data that does not meet conversion rules will become dirty data and be skipped.

NOTE

- When importing data from a relational database to HDFS or OBS, you do not need to configure data conversion. Data is separated by commas (,) and saved to HDFS or OBS.
- When exporting data from HDFS or OBS to a relational database, you do not need to configure data conversion. Data is separated by commas (,) and saved to the relational database.

Operator Description

Loader operators have three types:

- Input Operators

  First step of data conversion. This type of operator converts data into fields. Only one input operator can be used in each conversion. The input operator is mandatory in HBase or Hive data import and export.

- Conversion Operators

  Intermediate conversion step of data conversion. This type of operator is optional. The conversion operators can be used together in any combination. Conversion operators can process only fields. Therefore, an input operator must be used first to convert data into fields.

- Output Operators

  Last step of data conversion. Only one output operator can be used in each conversion for exporting processed fields. The output operator is mandatory in HBase or Hive data import and export.

Table 20-96 List of operator types

Node Type	Description
Input:	• CSV file input: Each line in the file is converted into multiple input fields based on the specified delimiter. • Fixed-width file input: Each line of the file is converted into multiple input fields based on the characters or bytes with configurable length. • Table input: converts specified columns in a relational database table into input fields of the same quantity. • HBase input: converts specified columns in an HBase table into input fields of the same quantity. • HTML input: converts elements in an HTML file into input fields. • Hive input: converts specified columns in a Hive table into input fields of the same quantity.

Node Type	Description
Convert	• Long integer to time conversion: Implements the conversion between long integer values and date types. • Null value conversion: Replaces a null value with a specified value. • Constant field adding: Generates a constant field. • Random value conversion: generates a random number field. • Concatenation and conversion: concatenates existing fields to generate new fields. • Delimiter conversion: separates existing fields with specified separators to generate new fields. • Modulo conversion: performs modulo operation on an existing field to generate a new field. • Character string cutting: cuts existing string fields by the specified start position and end position to generate new fields. • EL operation conversion: specifies a calculator to calculate field values. Currently, the following operators are supported: md5sum, sha1sum, sha256sum, and sha512sum. • Character string case conversion: converts the upper and lower cases of existing fields to generate new fields. • Character string reverse conversion: reverses existing character string fields to generate new fields. • Character string space clearing and conversion: clears the spaces on the left and right of the existing character string fields to generate new fields. • Row filtering conversion: filters rows that contain triggering conditions by configuring logic conditions. • Domain update: updates fields values when certain conditions are met.
Output:	• Hive Output: exports existing fields to a Hive table. • Table output: exports existing fields to a relational database table. • File output: uses delimiters to concatenate existing fields and exports new fields to a file. • HBase Output: exports existing fields to an HBase table.

Field Description

Fields in the job configuration are data items defined by Loader based on service requirements to match user data. Fields have specific types and the fields types must be consistent with the actual user data types.

20.8.2 Loader Input Operators

20.8.2.1 CSV File Input

Overview

The **CSV File Input** operator imports all files that can be opened by using a text editor.

Input and Output

- Input: test files
- Output: fields

Parameter Description

Table 20-97 Operator parameter description

Parameter	Description	Type	Mandatory	Default Value
Delimiter	Delimiter in a CSV file for separating data lines.	string	Yes	,
Line Delimiter	Line delimiter, which can be any string specified by users based on the actual situation. The OS line delimiter is used by default.	string	No	\n
Filename as field	User-defined field whose value is the name of the file that stores the current data.	string	No	None
Absolute path	Indicates whether the file name used as the value of **Filename as field** contains an absolute path. Selecting the option button indicates that the file name contains an absolute path; deselecting the option button indicates that the file name does not contain a path.	boolean	No	Deselect

Parameter	Description	Type	Mandatory	Default Value
Validate input field	Checks whether the input field matches the value type. If the value is **NO**, no check is performed. If the value is **YES**, whether the input field matches the value type is checked. If the input fields do not match the value type, the line is skipped.	enum	Yes	YES
Input fields	Information about input fields: • **position**: Position of the field after data lines in the source file are separated by delimiters. The position sequence starts from 1. • **field name**: Field name. • **type**: Field type. • **date format**: If the field type is **DATE**, **TIME**, or **TIMESTAMP**, you must specify a time format. If the field type is set to other values, the time format is invalid. An example time format is **yyyyMMdd HH:mm:ss**. • **length**: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length.	map	Yes	None

Data Processing Rule

- Each data line is separated into multiple fields by using delimiters and the fields are used by the subsequent conversion operator.
- If the field value does not match the actual type, the data in the line will become dirty data.

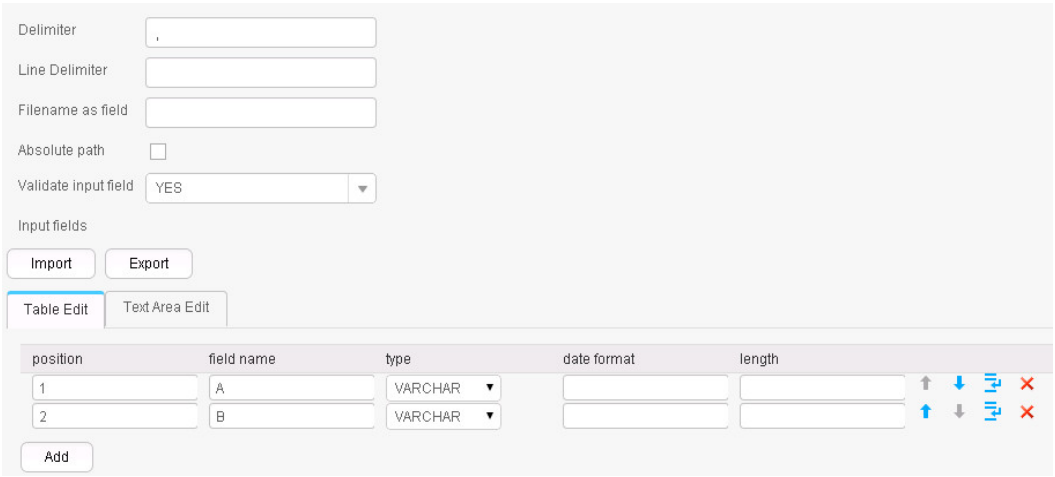
- If the number of input field columns is equal to the number of field columns actually included in the original data, the data in the line will become dirty data.

Example

The following figure shows the source file.

```
2016,year
year,2016
```

Configure the **CSV File Input** operator, set **Delimiter** to a comma (**,**), and generate fields A and B.



Fields A and B are generated, as shown in the following figure.

```
2016,year
year,2016
```

20.8.2.2 Fixed File Input

Overview

The **Fixed File Input** operator converts each line in a file into multiple fields by character or byte of a configurable length.

Input and Output

- Input: text file
- Output: fields

Parameter Description

Table 20-98 Operator parameter description

Parameter	Description	Type	Mandatory	Default Value
Line Delimiter	Line delimiter, which can be any string specified by users based on the actual situation. The OS line delimiter is used by default.	string	No	\n
Fixed length unit	Length unit. The options are **char** and **byte**.	enum	Yes	char
Input fields	Information about input fields: • **fixed length**: Field length. The ending of the first field is the starting of the second field, the ending of the second field is the starting of the third field, and so on. • **field name**: Names of input fields. • **type**: Field type. • **date format**: If the field type is **DATE**, **TIME**, or **TIMESTAMP**, you must specify a time format. If the field type is set to other values, the time format is invalid. An example time format is **yyyyMMdd HH:mm:ss**. • **length**: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length.	map	Yes	None

Data Processing Rule

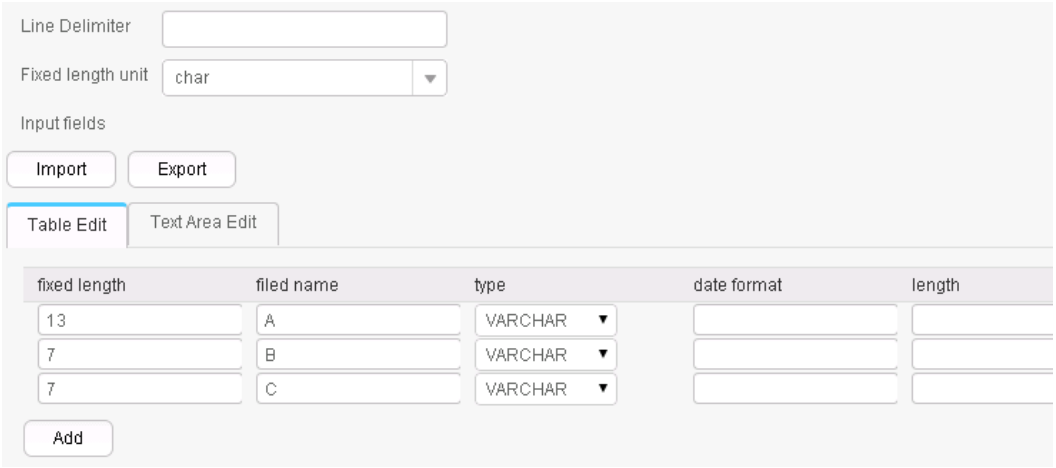
- The source file is split based on the input field length to generate fields.
- If the field value does not match the actual type, the data in the line will become dirty data.
- If the field split length is greater than the length of the original field value, the data split fails and the line becomes dirty data.

Example

The following figure shows the source file.

```
fusionInsightbigdataproduct
```

Configure the **Fixed File Input** operator to generate fields A, B, and C.



The three fields are generated, as shown in the following figure.

```
fusionInsight,bigdata,product
```

20.8.2.3 Table Input

Overview

Table Input operator converts specified columns in a relational database table into input fields of the same quantity.

Input and Output

- Input: table columns
- Output: fields

Parameter Description

Table 20-99 Operator parameters description

Parameter	Description	Type	Mandatory	Default Value
Input fields	Information about relational database input fields: • **position**: position of input fields • **field name**: input field name • **type**: field type • length: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length.	map	Yes	None

Data Processing Rule

- Fields are generated in a specified order. Table columns to be converted are specified by **From** in step 2 of job configuration. If **Table column names** is set, the value is the table columns to be converted; if **Table column names** is not set, the table columns to be converted are all table columns in the table by default or the columns specified by the query conditions set by **Table SQL statement**.
- The number of input fields cannot be greater than number of specified columns; otherwise, all data becomes dirty data.
- If the field value does not match the actual type, the data in the line will become dirty data.

Example

Use SQL Server 2014 as an example. Run the following command to create a **test** table:

create table test (id int, name text, value text);

Insert three data lines to the test table:

insert into test values (1,'zhangshan','zhang');

insert into test values (2,'lisi','li');

insert into test values (3,'wangwu','wang');

Query the table:

	id	name	value
1	1	zhangshan	zhang
2	2	lisi	li
3	3	wangwu	wang

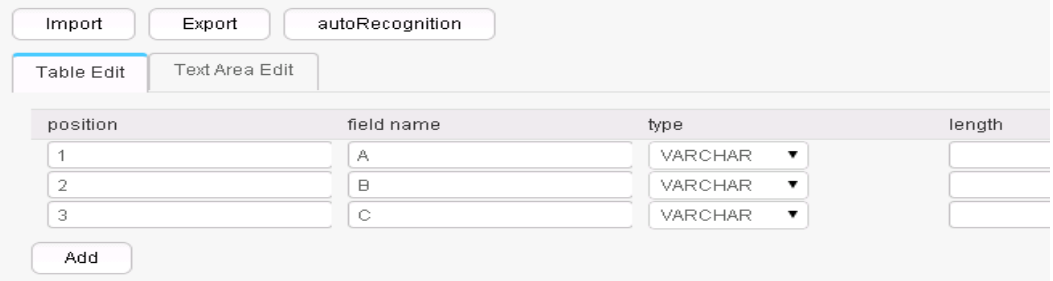
Configure the **Table Input** operator to generate the following fields:

After the data connector is set, click **Automatic Identification**. The system will automatically read fields in the database and select required fields for adding. You only need to optimize or modify the fields manually based on service scenarios.

NOTE

- This operation will overwrite existing data in the table.
- After you click **autoRecognition**, manually check the field types automatically identified by the system to ensure that they are consistent with the actual ones in the table.

  For example, the system automatically identifies the **date** type in the Oracle database as the **timestamp** type. If you do not manually change the type, an error will be reported when data is queried in the Hive table.



Configure the output operator to output data to HDFS or OBS. The result is as follows:

```
1,zhangshan,zhang
2,lisi,li
3,wangwu,wang
```

20.8.2.4 HBase Input

Overview

The **HBase Input** operator converts specified columns in an HBase table into input fields of the same quantity.

Input and Output

- Input: HBase table columns
- Output: fields

Parameter Description

Table 20-100 Operator parameter description

Parameter	Description	Type	Mandatory	Default Value
Hbase Table Type	HBase table type. The options include **normal** (common HBase table) and **phoenix**.	enum	Yes	normal
HBase table name	HBase table name. Only one HBase table is supported.	string	Yes	None
HBase input fields	HBase input information: • **family name**: HBase column family name. • **column name**: HBase column name. • **field name**: Names of input fields. • **type**: Field type. • **length**: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length. • **is rowkey**: Indicates whether a column is a primary key column. A common HBase table can have only one primary key, while a phoenix table can have multiple primary keys. If multiple primary keys are configured, they are combined according to the configuration sequence. At least one primary key column must be configured.	map	Yes	None

Data Processing Rule

- If the HBase table name does not exist, the job fails to be submitted.
- If the configured column names are inconsistent with the HBase table column names, the data cannot be read and the number of imported data records is 0.
- If the number of input field columns is greater than the number of field columns actually included in the original data, all data becomes dirty data.
- If the field value does not match the actual type, the data in the line will become dirty data.

Example

Use the data export from HBase to sqlserver2014 as an example.

In sqlserver2014, run the following statement to create an empty data test_1 for storing HBase data:

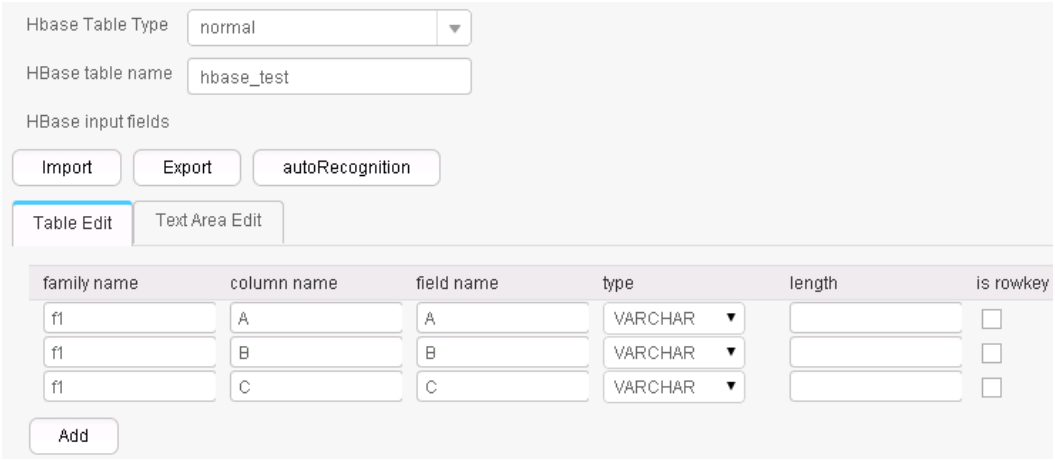
create table test_1 (id int, name text, value text);

Configure the **HBase Input** operator to generate fields A, B, and C.

After the database connection is set up, click **autoRecognition**. The system will automatically read fields in the database and select required fields for adding. You only need to optimize or modify the fields manually based on service scenarios.

NOTE

Performing this operation will overwrite existing data in the table.



Use the **Table Out** operator to export A, B, and C to the test_1 table.

select * from test_1;

	id	name	value
1	1	zhangshan	zhang
2	2	lisi	li
3	3	wangwu	wang

20.8.2.5 HTML Input

Overview

HTML Input operator imports a regular HTML file and converts elements in the HTML file into input fields.

Input and Output

Input: HTML file

Output: multiple fields

Parameter Description

Table 20-101 Operator parameters description

Parameter	Description	Type	Mandatory	Default Value
parent tag	Upper-layer HTML tag of all fields for limiting the search scope.	string	Yes	None
Filename as field	User-defined field whose value is the name of the file that stores the current data.	string	No	None
Absolute file name	Whether the file name used as the value of **Filename as field** contains an absolute path. Selecting the option button indicates that the file name contains an absolute path; deselecting the option button indicates that the file name does not contain a path.	boolean	No	No
Validate input field	Whether to check the type matching between the input field and the value. If the value is **NO**, the field is not checked. If the value is **YES**, the field will be checked. If the input field does not match the value type, the line is skipped.	enum	Yes	YES

Parameter	Description	Type	Mandatory	Default Value
Input fields	Information about input fields: • **position**: Position of the field. The position sequence starts from 1. • **field name**: field name • **field tag**: field tag • **keyword**: A keyword can be configured to match the content of the tag. Wildcards are supported. For example, if the tag content is **name**, you can configure the keyword ***name***. • **type**: field type • **date format**: If the field type is **DATE**, **TIME**, or **TIMESTAMP**, you need to specify the time format. If the field type is neither of them, the time format is invalid. The example time format is **yyyyMMdd HH:mm:ss**. • **length**: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length.	map	Yes	None

Data Processing Rule

- **parent tag** is configured first to limit the search scope. The value of **parent tag** must exist; otherwise, the obtained content is empty.
- **Input fields** are configured so that the sub-tags can be used to precisely locate the tags of fields. If the tags are the same, keywords will be used for precise matching.
- The keyword is used to match the content of the field. The configuration method is similar to that of the **File filter** field in the **From** settings. The wildcard **(*)** is supported. The following three tags are provided to assist in locating the field:

  a. **#PART**: indicates the values matched by wildcard *. If there are multiple *, you can specify an order from left to right and obtain content that matches the sequence number *. For example, **#PART1** indicates to

obtain the value that matches the first * and **#PART8** indicates to obtain the value that matches the eighth *).

b. **#NEXT**: indicates that you can obtain the value next to the value that matches the tag.

c. **#ALL**: indicates that you can obtain all the values that match the tag.

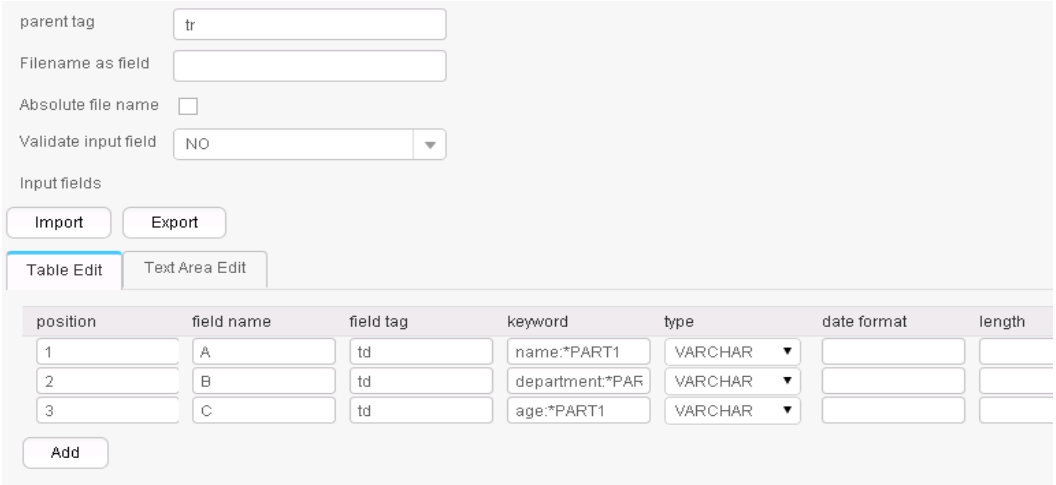
- If the tag is configured incorrectly, the obtained value is empty, but no error is reported.

Example

The following figure shows the source file.

```
<html>
<body>
<table>
<tr>
<td>name:zhangshan</td>
<td>department:FusionInght</td>
<td>age:25</td>
</tr>
</table>
</body>
</html>
```

Configure the **HTML Input** operator to generate fields A, B, and C.



Three fields are generated, as shown in the following figure.

```
zhangshan,FusionInght,25
```


20.8.2.6 Hive input

Overview

The **Hive Input** operator converts specified columns in an HBase table into input fields of the same quantity.

Input and Output

- Input: Hive table columns
- Output: fields

Parameters

Table 20-102 Operator parameters description

Parameter	Description	Node Type	Mandatory	Default Value
Hive database	Name of a Hive database	String	No	default
Hive table name	Name of the Hive table configured Only one Hive table is supported.	String	Yes	None.
Partition filter	Configures the partition filter can export data of specific partitions. The parameter is null by default and data of the whole table can be exported. For example, to export data of a table whose partition field's locale value is **CN** or **US**, the input is as follows: **locale = "CN" or locale = "US"**	String	No	-
Hive input field	Configures the input information of Hive ● column name: Hive column name. ● field name: Input field name. ● type: Field type. ● length: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length.	map	Yes	-

Data Processing Rule

- If the Hive table name does not exist, the job fails to be submitted.
- If the configured column names are inconsistent with the Hive table column names, the data cannot be read and the number of imported data records is 0.
- If the field value does not match the actual type, the data in the line will become dirty data.

Example

Use the data export from Hive to SQL Server 2014 as an example.

In SQL Server 2014, run the following statement to create an empty table **test_1** for storing Hive data. Run the following statement:

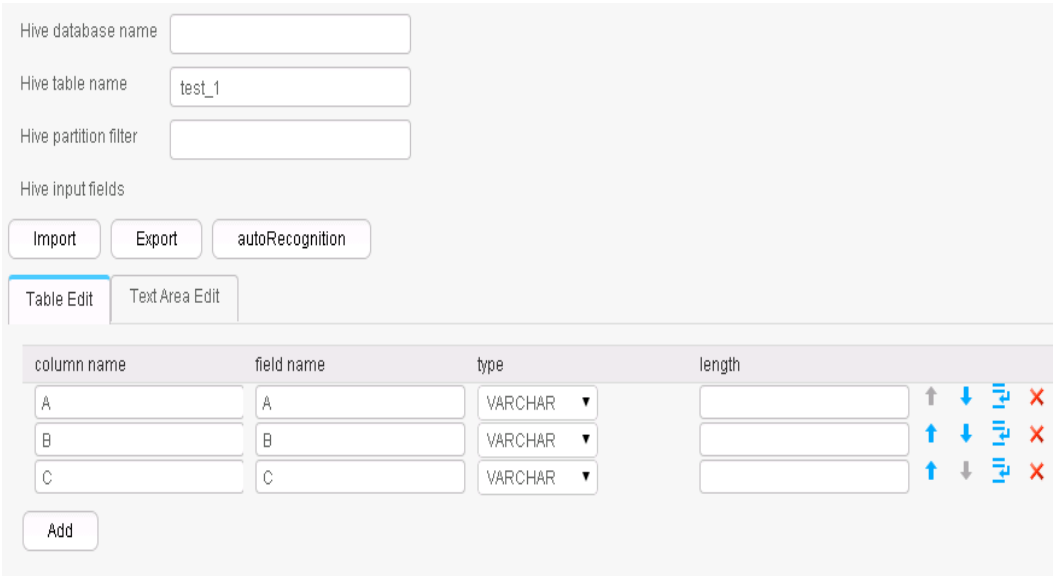
create table test_1 (id int, name text, value text);

Configure the **Hive Input** operator to generate fields A, B, and C.

After the data connector is set, click **Automatic Identification**. The system will automatically read fields in the database and select required fields for adding. You only need to optimize or modify the fields manually based on service scenarios.

NOTE

Performing this operation will overwrite existing data in the table.



Use the **Table Out** operator to export A, B, and C to the **test_1** table.

select * from test_1;

	id	name	value
1	1	zhangshan	zhang
2	2	lisi	li
3	3	wangwu	wang

20.8.2.7 Spark Input

Overview

The **Spark Input** operator converts specified columns in an SparkSQL table into input fields of the same quantity.

Input and Output

- Input: SparkSQL table column
- Output: fields

Parameters

Table 20-103 Operator parameters description

Parameter	Description	Type	Mandatory	Default Value
Spark database	Name of a Spark SQL database	String	No	default
Spark table name	Configures the SparkSQL table name. Only one SparkSQL table is supported.	String	Yes	None
Partition filter	Configures the partition filter can export data of specific partitions. The parameter is null by default and data of the whole table can be exported. For example, to export data of a table whose partition field's locale value is **CN** or **US**, the input is as follows: **locale = "CN" or locale = "US"**	String	No	-

Parameter	Description	Type	Mandatory	Default Value
Input fields of Spark	Configures the input information of SparkSQL ● column name: SparkSQL column name. ● field name: Input field name. ● type: Field type. ● length: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length.	map	Yes	-

Data Processing Rule

- If the SparkSQL table name does not exist, the job fails to be submitted.
- If the configured column names are inconsistent with the SparkSQL table column names, the data cannot be read and the number of imported data records is 0.
- If the field value does not match the actual type, the data in the line will become dirty data.

Example

Use the data export from Spark to SQL Server 2014 as an example.

In SQL Server 2014, run the following statement to create an empty table **test_1** for storing SparkSQL data. Run the following statement:

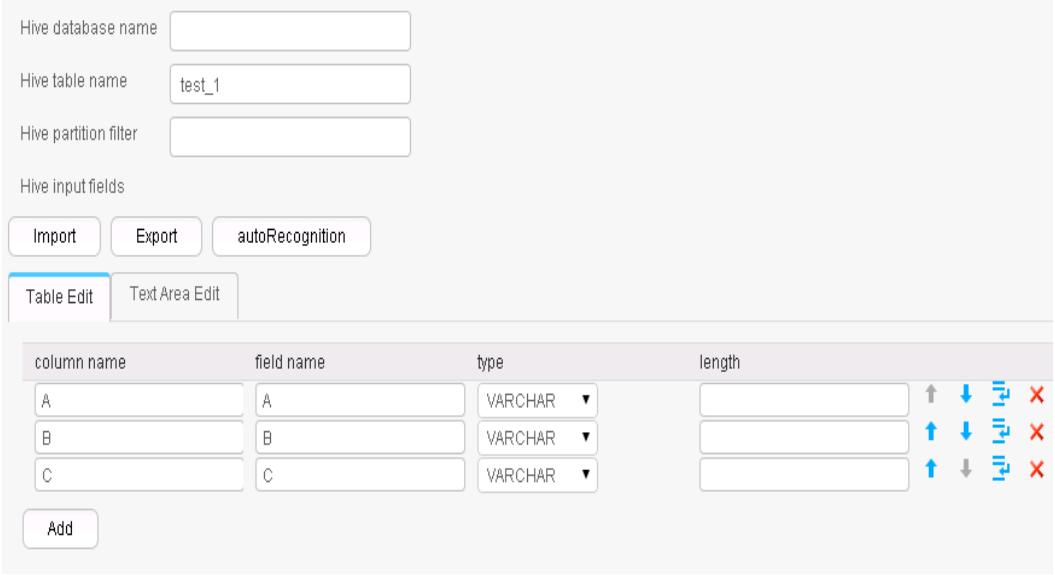
create table test_1 (id int, name text, value text);

Configure the **Spark Input** operator to generate fields A, B, and C.

After the data connector is set, click **Automatic Identification**. The system will automatically read fields in the database and select required fields for adding. You only need to optimize or modify the fields manually based on service scenarios.

NOTE

Performing this operation will overwrite existing data in the table.



Use the **Table Out** operator to export A, B, and C to the **test_1** table.

select * from test_1;

	id	name	value
1	1	zhangshan	zhang
2	2	lisi	li
3	3	wangwu	wang

20.8.3 Loader Conversion Operators

20.8.3.1 Long Date Conversion

Overview

The **Long Date Conversion** operator performs long integer and date conversion.

Input and Output

- Input: fields to be converted
- Output: new fields

Parameter Description

Table 20-104 Operator parameter description

Parameter	Description	Type	Mandatory	Default Value
convert type	Types of long integer and date conversion: • **long to date**: converts long integers to date. • **long to time**: converts long integers to time. • **long to timestamp**: converts long integers to timestamp. • **date to long**: converts date to long integers. • **time to long**: converts time to long integers. • **timestamp to long**: converts timestamp to long integers.	enum	Yes	long to date
input field name	Name of input fields to be converted. Set this parameter to the names of fields generated in the previous conversion step.	string	Yes	None
output field name	Names of output fields.	string	Yes	None
field unit	Unit of a long integer field. According to **convert type**, the value is an input field or generated field. The options are **second** and **millisecond**.	enum	Yes	second
output field type	Output field type. The options are **BIGINT**, **DATE**, **TIME**, and **TIMESTAMP**.	enum	Yes	BIGINT
date format	Time field format, for example, **yyyyMMdd HH:mm:ss**.	string	No	None

Data Processing Rule

- If the original data includes null values, no conversion is performed.
- If the number of input field columns is greater than the number of field columns actually included in the original data, all data becomes dirty data.
- If a type conversion error occurs, the current data is saved as dirty data.

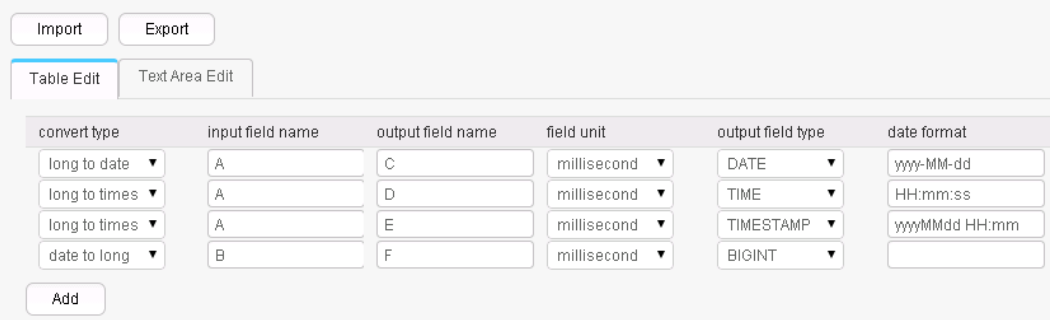
Example

Use the **CSV File Input** operator to generate fields A and B.

The following figure shows the source file.

```
1453431755874,2016-01-22 10:40:00
```

Configure the **Long Date Conversion** operator to generate four new fields C, D, E, and F. Their types are DATE, TIME, TIMESTAMP, and BIGINT, respectively.



The following figure shows the output of the conversion.

```
1453431755874,2016-01-22,2016-01-22,11:02:35,20160122 11:02:35,1453430400000
```

20.8.3.2 Null Value Conversion

Overview

The **null value conversion** operator replaces null values with specified values.

Input and Output

- Input: fields with null values
- Output: original fields with new values

Parameter Description

Table 20-105 Operator parameters description

Parameter	Description	Node Type	Mandatory	Default Value
Input field name	Names of fields that may have null values. Set this parameter to the names of existing fields.	string	Yes	None
Replace by this value	Specified values for replacing null values.	string	Yes	None

Data Processing Rule

When field values are empty, specified values are added.

Example

Use the **CSV File Input** operator to generate two fields A and B.

The following figure shows the source file.

```
,value1
key2,value2
key3,
```

Configure the **null value conversion** operator, as shown in the following figure.

After replacement, the values of fields A and B are as follows:

```
newKey,value1
key2,value2
key3,newValue
```

20.8.3.3 Constant Field Addition

Overview

The **Add Constants** operator generates constant fields.

Input and Output

- Input: none
- Output: constant fields

Parameter Description

Table 20-106 Operator parameters description

Parameter	Description	Type	Mandatory	Default Value
Constant fields	Information about constant fields: • **output field name**: Names of the configured fields. • **type**: field type • **date format**: If the field type is **DATE**, **TIME**, or **TIMESTAMP**, you need to specify the time format. If the field type is neither of them, the time format is invalid. The example time format is **yyyyMMdd HH:mm:ss**. • **length**: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length. • **constant value**: Constant value of the correct type.	map	Yes	None

Data Processing Rule

This operator generates constant fields of the specified type.

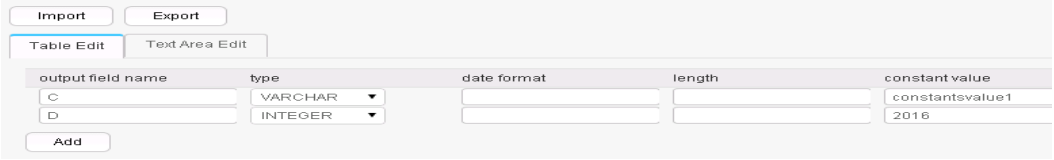
Example

Use the **CSV File Input** operator to generate two fields A and B.

The following figure shows the source file.

```
,value1
key2,value2
key3,
```

Configure the **Add Constants** operator to add fields C and D.



After adding the constants, fields A, B, C, and D are generated, as shown in the following figure.

```
,value1,constantsvalue1,2016
key2,value2,constantsvalue1,2016
key3,,constantsvalue1,2016
```

20.8.3.4 Random Value Conversion

Overview

Generate Random operator configures new values as random value fields.

Input and Output

- Input: none
- Output: random value fields

Parameter Description

Table 20-107 Operator parameters description

Parameter	Description	Type	Mandatory	Default Value
output field name	Names of generated random value fields.	string	Yes	None
length	Field length.	map	Yes	None
type	Field type. The options are **VARCHAR**, **INTEGER**, and **BIGINT**.	enum	Yes	VARCHAR

Data Processing Rule

The operator generates random value fields of specified type.

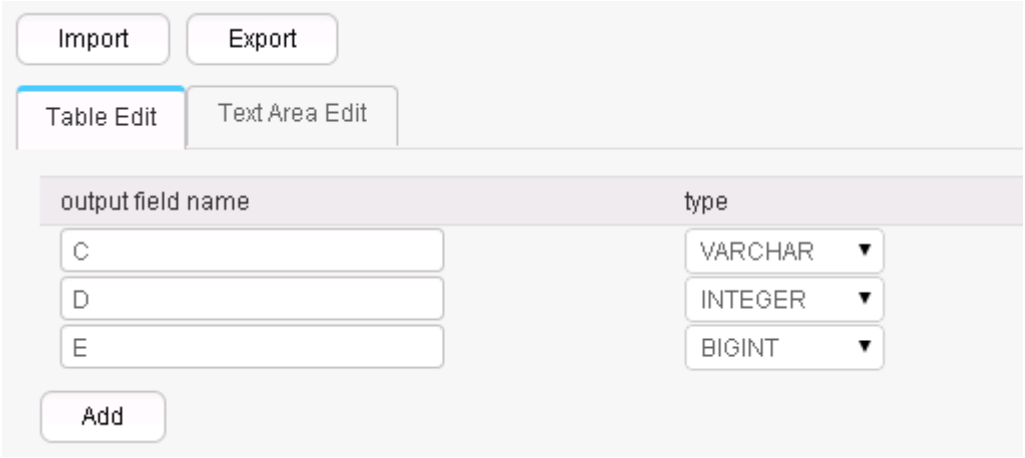
Example

Use the **CSV File Input** operator to generate two fields A and B.

The following figure shows the source file.


```
,value1
key2,value2
key3,
```

Configure the random value conversion operator to generate fields C, D, and E.



Five fields are generated.

```
,value1,2druceak69ril,769974975,8452014577467885098
key2,value2,7oq2dku93q9cg,1631427868,867914116689501757
key3,,2jg5e7b1m17kq,654806209,2477823020516316030
```

The random value fields generated each time are different.

20.8.3.5 Concat Fields

Overview

The **Concat Fields** operator concatenates existing fields by using delimiters to generate new fields.

Input and Output

- Input: fields to be concatenated
- Output: new fields

Parameter Description

Table 20-108 Operator parameter description

Parameter	Description	Type	Mandatory	Default Value
Output field name	Name of a field generated after concatenation.	string	Yes	None
Delimiter	Concatenation character. The value can be blank.	string	No	Empty string
Fields to be merged	Names of fields to be concatenated. **field name** must be set to the names of fields generated in the previous conversion step. Multiple field names can be added.	map	Yes	None

Data Processing Rule

- Use delimiters to concatenate the fields specified by **Fields to be merged** in order and assign the output to **Output field name**.
- If the value of a field is null, the value is changed to an empty string and then concatenated with other field values.

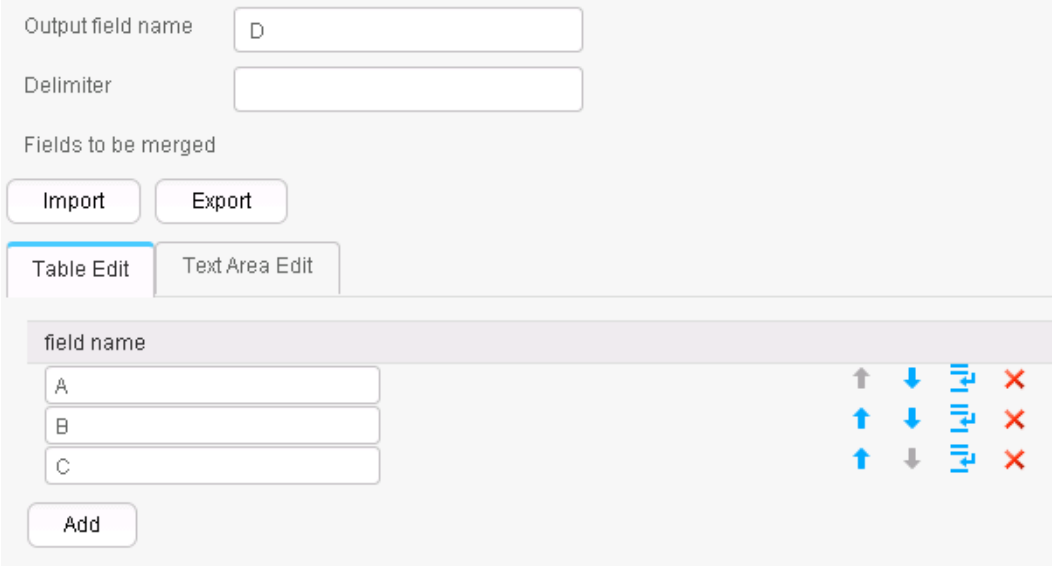
Example

Use the **CSV File Input** operator to generate fields A, B, and C.

The following figure shows the source file.

```
happy,new,year
welcome,to,2016
```

Configure the **Concat Fields** operator, set **Delimiter** to blank space, and generate field D.



After concatenation, fields A, B, C, and D are generated, as shown in the following figure.

```
happy,new,year,happy new year
welcome,to,2016,welcome to 2016
```

20.8.3.6 Extract Fields

Overview

The **Extract Fields** separates an existing field by using delimiters to generate new fields.

Input and Output

- Input: field to be separated
- Output: new fields

Parameter Description

Table 20-109 Operator parameter description

Parameter	Description	Type	Mandatory	Default Value
Input field name	Name of a field to be separated. Set this parameter to the name of a field generated in the previous conversion step.	string	Yes	None
Delimiter	Delimiter.	string	Yes	None

Parameter	Description	Type	Mandatory	Default Value
Fields extracted	Fields generated after field separation. Multiple fields can be generated after field separation. • **position**: Position of fields generated after field separation. • **output field name**: Names of output fields.	map	Yes	None

Data Processing Rule

- The value of the input field is separated by specified delimiters and the segments are assigned to the new fields.
- If the number of field columns after separation is greater than the actual number allowed by the original data, the line will become dirty data.

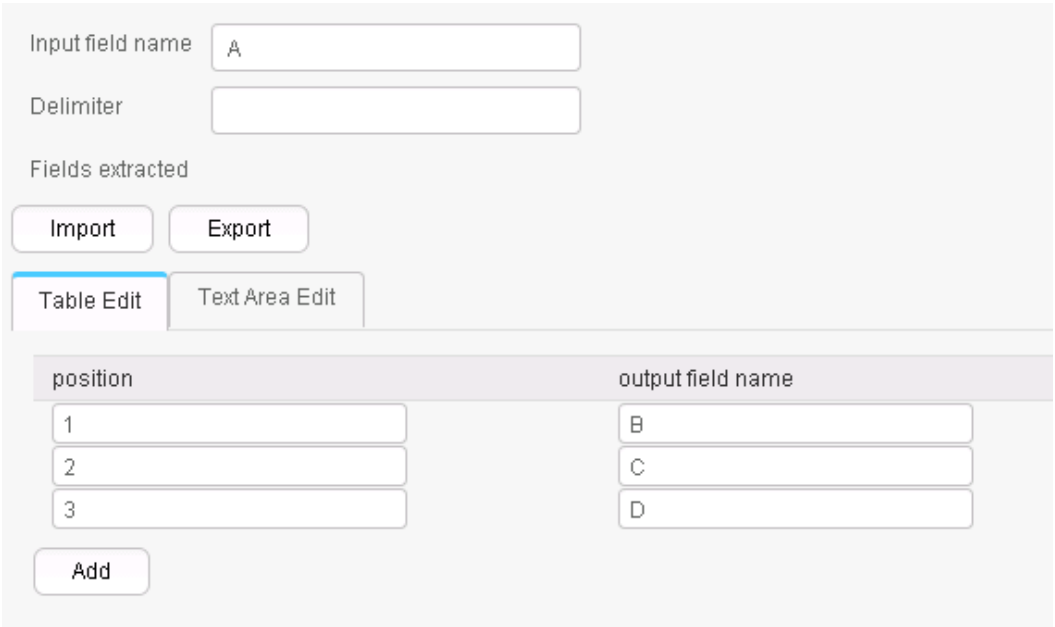
Example

Use the **CSV File Input** operator to generate field A.

The following figure shows the source file.

```
happy new year
welcome to 2016
```

Configure the **Extract Fields** operator, set **Delimiter** to blank space, and generate three fields B, C, and D.



After conversion, fields A, B, C, and D are generated, as shown in the following figure.

```
happy new year,happy,new,year
welcome to 2016,welcome,to,2016
```

20.8.3.7 Modulo Integer

Overview

The **Modulo Integer** operator performs modulo operations on integer fields to generate new fields.

Input and Output

- Input: integer fields
- Output: new fields

Parameter Description

Table 20-110 Operator parameter description

Parameter	Description	Type	Mandatory	Default Value
Modulo fields	Modulo operation information: • **input field name**: Names of input fields. Set this parameter to the names of fields generated in the previous conversion step. • **output field name**: Names of output fields. • **modulus**: Values used for a modulo operation.	map	Yes	None

Data Processing Rule

- The operator generates new fields and the values are those after the modulo operation.
- The field values must be integers; otherwise, the current line becomes dirty data.

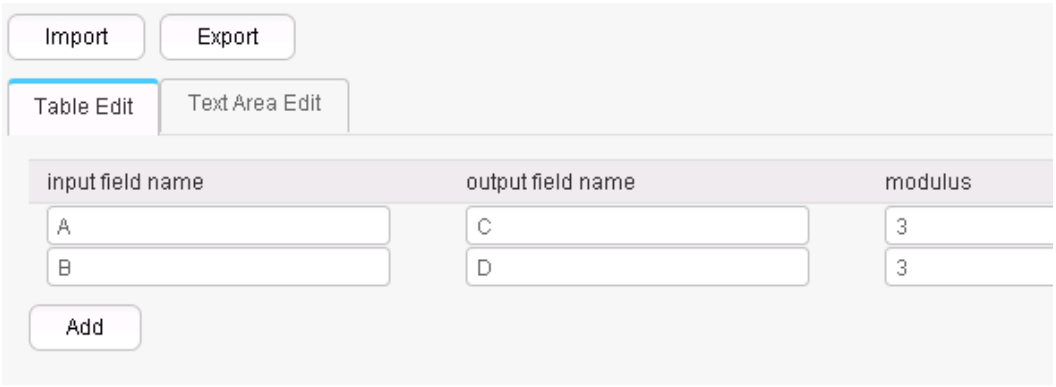
Example

Use the **CSV File Input** operator to generate fields A and B.

The following figure shows the source file.


```
10,12
2015,2016
```

Configure the **Modulo Integer** operator to generate two new fields C and D.



After the modulo operation, fields A, B, C, and D are generated, as shown in the following figure.

```
10,12,1,0
2015,2016,2,0
```

20.8.3.8 String Cut

Overview

The **String Cut** operator cuts existing fields to generate new fields.

Input and Output

- Input: fields to be cut
- Output: new fields

Parameter Description

Table 20-111 Operator parameter description

Parameter	Description	Type	Mandatory	Default Value
Fields to be cut	Information about a cut field: • **input field name**: Names of input fields. Set this parameter to the names of fields generated in the previous conversion step. • **output field name**: Names of output fields. • **start position**: Cutting start position, starting from sequence 1. • **end position**: Cutting end position. If the length of the cut string cannot be determined, you can set this parameter to -1, indicating the end of a string to be cut. • **output field type**: Type of output fields. • **output field length**: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **output field type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **output field type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length.	map	Yes	None

Data Processing Rule

- **start position** and **end position** are used to cut the original fields and generate new fields.
- If **end position** is set to **-1**, the end of a string is to be cut. In other cases, the value of **end position** must be greater than the value of **start position**.
- If the value of **start position** or **end position** is greater than the length of the input field, the line will become dirty data.

Example

Use the **CSV File Input** operator to generate fields A and B.

The following figure shows the source file.

```
abcd,product
FusionInsight,Bigdata
```

After configuring the **String Cut** operator, fields C and D are generated.

After cutting, the following fields are generated.

```
abcd,product,abc,prod
FusionInsight,Bigdata,Fus,Bigd
```

20.8.3.9 EL Operation

Overview

The **EL Operation** operator calculates field values and generates new fields. The algorithms that are currently supported include md5sum, sha1sum, sha256sum, and sha512sum.

Input and Output

- Input: fields to be converted
- Output: fields generated after the EL expression conversion

Parameter Description

Table 20-112 Operator parameter description

Parameter	Description	Type	Mandatory	Default Value
Field generated by el operation	EL expression configuration: • **name**: Name of the expression output result. • **el expression**: Expression. The format is *expression name(input field name,value indicating whether to use lower case letters to indicate the output result)*, for example, md5sum(fieldname,true). – md5sum: generates md5 values. – sha1sum: generates sha1 values. – sha256sum: generates sha256 values. – sha512sum: generates sha512 values. • **type**: Type of the expression output result. **VARCHAR** is recommended. • **date format**: Format of the expression output result. • **length**: Length of the expression output result.	map	Yes	None

Data Processing Rule

- The operator calculates fields values and generates new fields.
- The type of the new fields can only be VARCHAR.

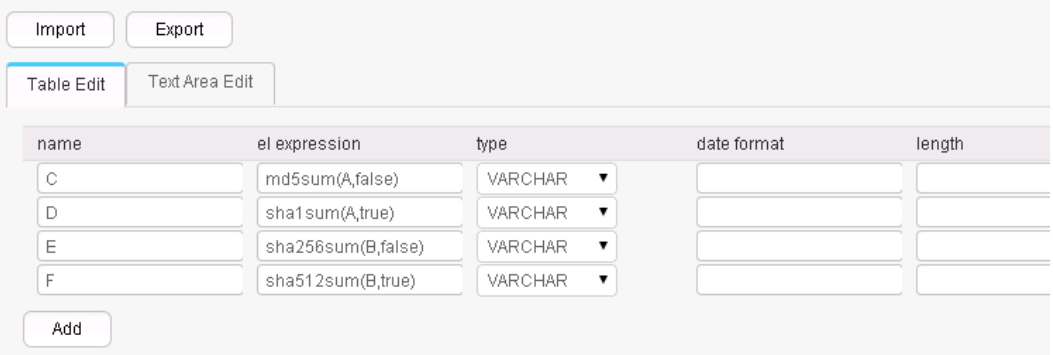
Example

Use the **CSV File Input** operator to generate fields A and B.

The following figure shows the source file.

```
2016,year
year,2016
```

Configure the **EL Operation** operator to generate fields C, D, E, and F.



Six fields are generated, as shown in the following figure.

```
2016,year,95192C98732387165BF8E396C0F2DAD2,ab39c54239118a4b086b878b7878100f769dd1
97,4CB4EA25583C25647247AE96FC90225D99AD7A6FABC3E2C2FD13C502E323CD9E,779edfe0463b2
596e7a83e4c59083e19242e8c51eace8e2ec57704643be5e15ba80f79af227cf3ea2e2362b4081377
96a1d82cb0535652b99844bb9a62019563
year,2016,84CDC76CABF41BD7C961F6AB12F117D8,4ff0b1538469338a0073e2cdaab6a517801b6a
b4,DA6E2F539726FABD1F8CD7C9469A22B36769137975B28ABC65FE2DC29E659B77,da0ae9104086a
1c58f89f82766ac55a02c8ab44277ce39f959ec0e73391bef651c6f9793657396ce47fbd846068465
ccbf3056764424bed9be7789bd1101ace7
```

20.8.3.10 String Operations

Overview

The **String Operations** operator converts the upper and lower cases of existing fields to generate new fields.

Input and Output

- Input: fields whose case is to be converted
- Output: new fields after conversion

Parameter Description

Table 20-113 Operator parameter description

Parameter	Description	Type	Mandatory	Default Value
Fields to be processed	Information about fields for string case conversion: • **input field name**: Names of input fields. Set this parameter to the names of fields generated in the previous conversion step. • **output field name**: Names of output fields. • **lower/upper**: Indicates whether data is to be converted into uppercase letters or lowercase letters.	map	Yes	None

Data Processing Rule

- Case conversion is performed for strings.
- If the input data is null, no case conversion is performed.

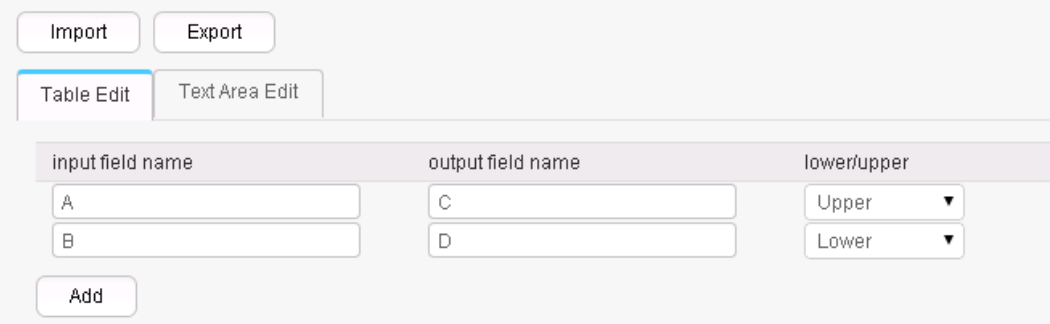
Example

Use the **CSV File Input** operator to generate fields A and B.

The following figure shows the source file.

```
abcd,product
FusionInsight,Bigdata
```

After configuring the **String Operations** operator, fields C and D are generated.



After conversion, four fields are generated, as shown in the following figure.

```
abcd,product,ABCD,product
FusionInsight,Bigdata,FUSIONINSIGHT,bigdata
```


20.8.3.11 String Reverse

Overview

The **String Reverse** operator reverses existing fields to generate new fields.

Input and Output

- Input: fields to be reversed
- Output: new fields

Parameter Description

Table 20-114 Operator parameter description

Parameter	Description	Type	Mandatory	Default Value
Fields to be reversed	Information about fields for string reversal conversion: • **input field name**: Names of input fields. Set this parameter to the names of fields generated in the previous conversion step. • **output field name**: Names of output fields. • **type**: Field type. • **out field length**: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length.	map	Yes	None

Data Processing Rule

- Value reversal conversion is performed for fields.
- If the input data is null, no reversal conversion is performed.
- If the number of input field columns is greater than the number of field columns actually included in the original data, all data becomes dirty data.

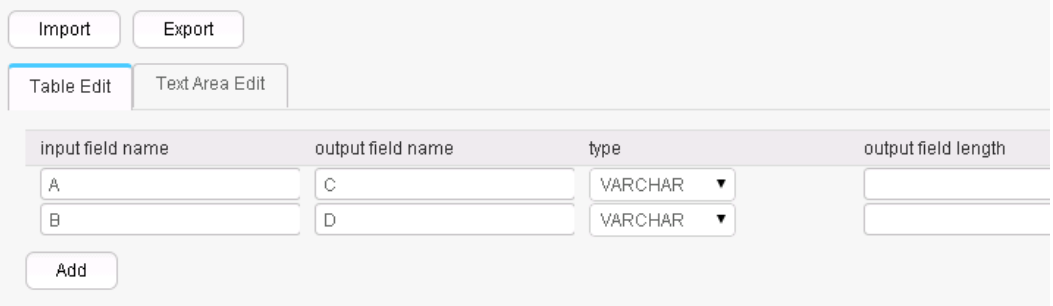
Example

Use the **CSV File Input** operator to generate fields A and B.

The following figure shows the source file.

```
abcd,product
FusionInsight,Bigdata
```

Configure the **String Reverse** operator to generate two new fields C and D.



After conversion, four fields are generated, as shown in the following figure.

```
abcd,product,dcba,tcudorp
FusionInsight,Bigdata,thgisnInoisuF,atadgiB
```

20.8.3.12 String Trim

Overview

The **String Trim** operator clears spaces contained in existing fields to generate new fields.

Input and Output

- Input: fields whose spaces are to be cleared
- Output: new fields

Parameter Description

Table 20-115 Operator parameter description

Parameter	Description	Type	Mandatory	Default Value
Fields to be trimmed	Information about fields for clearing spaces contained in strings: • **input field name**: Names of input fields. Set this parameter to the names of fields generated in the previous conversion step. • **output field name**: Names of output fields. • **trim type**: Space clearing mode (clearing starting spaces, ending spaces, or starting and ending spaces).	map	Yes	None

Data Processing Rule

- Clearing spaces at both ends of a value supports clearing spaces at the left end, at the right end, and at both ends.
- If the input data is null, no conversion is performed.
- If the number of input field columns is greater than the number of field columns actually included in the original data, all data becomes dirty data.

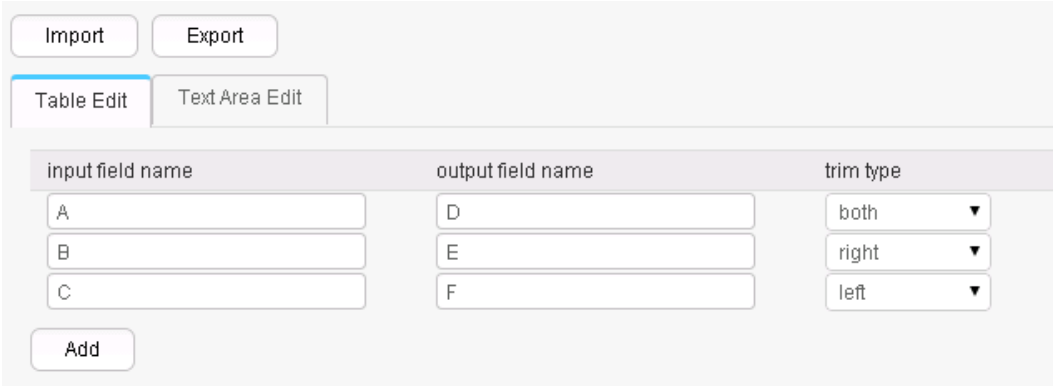
Example

Use the **CSV File Input** operator to generate fields A, B, and C.

The following figure shows the source file.

```
  welcome   ,to  ,  2016
happy  ,new , year
```

Configure the **String Trim** operator to generate three new fields D, E, and F.



Six fields are generated, as shown in the following figure.

```
  welcome   ,to  ,  2016,welcome,to,2016
happy  ,new , year,happy,new,year
```

20.8.3.13 Filter Rows

Overview

This **Filter Rows** operator filters rows that contain triggering conditions by configuring logic conditions.

Input and Output

- Input: fields used to create filter conditions
- Output: none

Parameter Description

Table 20-116 Operator parameters description

Parameter	Description	Type	Mandatory	Default Value
Condition logic connector	Condition logic connector. The options include **AND** and **OR**.	enum	Yes	AND
Conditions	Filter condition information: • **input field name**: Names of the input fields. Set this parameter to the names of the fields generated in the previous conversion step. • **operator**: Operator • **comparative value**. You can directly enter the value of a field referenced in the **#{Existing field name}** format.	map	Yes	None

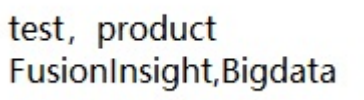
Data Processing Rule

- When the condition logic is **AND**, if no filtering condition is added, all data becomes dirty data; if the original data meets all the added filtering conditions, the current line becomes dirty data.
- When the condition logic is **OR**, if no filter condition is added, all data becomes dirty data; if the original data meets any of the added filter conditions, the current line becomes dirty data.

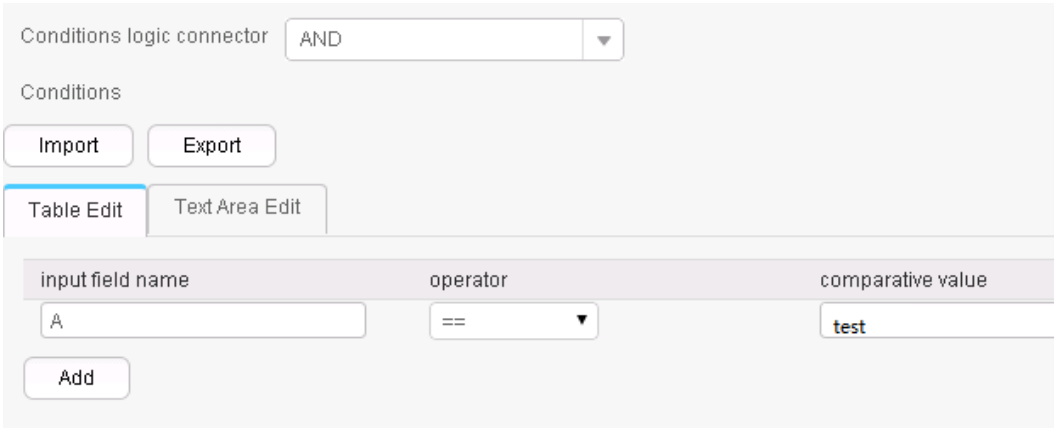
Example

Use the **CSV File Input** operator to generate two fields A and B.

The following figure shows the source file.



Configure the **Filter Rows** operator to filter out lines that contain **test**.



After the conversion, enter the original fields. The result is as follows:

```
FusionInsight,Bigdata
```

20.8.3.14 Update Fields Operator

Overview

The **Update Fields** operator updates fields values when certain conditions are met.

The types supported at present include **BIGINT**, **DECIMAL**, **DOUBLE**, **FLOAT**, **INTEGER**, **SMALLINT**, and **VARCHAR**. When the type is **VARCHAR** and the operator is **+**, strings will be added to the end of field values. The operator **-** is not supported. For other types, **+** and **-** indicate addition and subtraction of values. For all types, **=** indicates new value assignment.

Input and Output

Input: field

Output: input field

Parameter Description

Table 20-117 Operator parameters description

Parameter	Description	Type	Mandatory	Default Value
update field name	Fields to be updated	string	Yes	None
update operator	Operator, which can be **+**, **-**, or **=**.	enum	Yes	+
update value	Values to be updated	The type is the same as the field type.	No	None
Condition logic connector	Condition logic connector. The options include **AND** and **OR.**	enum	Yes	AND
Conditions	Filter condition information: • **input field name**: Names of the input fields. Set this parameter to the names of the fields generated in the previous conversion step. • **operator**: Operator • **comparative value**. You can directly enter the value of a field referenced in the **#{Existing field name}** format.	map	Yes	None

Data Processing Rule

- The operator checks whether conditions are met. If yes, the operator updates the field values. If no, the operator does not update the field values.
- If the field values are digits, the updated values are digits.
- If the fields are of the string type, the operator **-** cannot be used.

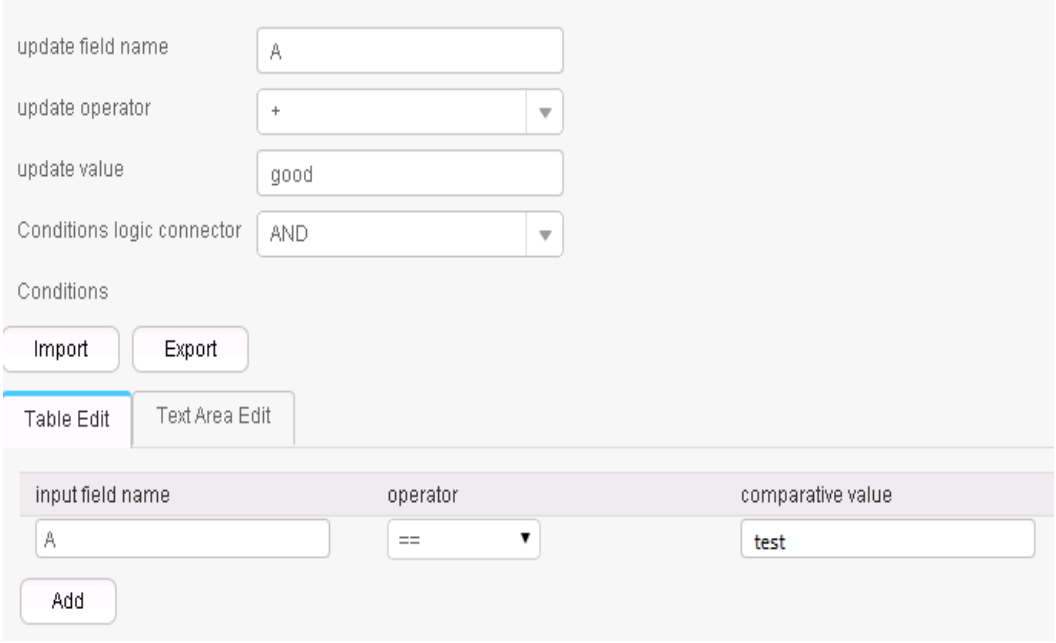
Example

Use the **CSV File Input** operator to generate two fields A and B.

The following figure shows the source file.

```
test，product
FusionInsight,Bigdata
```


Configure the **Update Fields** operator to update a value by adding **good** to the end of the value if the value is **test**.



The following figure shows the output result.

20.8.4 Loader Output Operators

20.8.4.1 Hive output

Overview

The **Hive Output** operator exports existing fields to specified columns of a Hive table.

Input and Output

- Input: fields to be exported
- Output: Hive table

Parameters

Table 20-118 Operator parameters description

Parameter	Description	Node Type	Mandatory	Default Value
Hive file storage format	Hive configuration file storage format. CSV, ORC, and RC are supported at present. **NOTE** • Parquet is a column-based storage format. In this format, the output field names of Loader be the same as the field names in Hive tables. • For Hive of versions later than 1.2.0, a field name, instead of field number, is used to parse ORC files. Therefore, the output field names of Loader must be the same as those in Hive tables.	enum	Yes	CSV
Hive file compression format	Hive table file compression format. Select a format from the drop-down list. If you select **NONE** or do not set this parameter, data is not compressed.	enum	Yes	NONE
Hive ORC file version	Version of the ORC file (when the storage format of the Hive table file is ORC).	enum	Yes	0.12
Output delimiter	Delimiter.	string	Yes	None.

Parameter	Description	Node Type	Mandatory	Default Value
Output fields	Information about output fields: • position: Position of output fields. • field name: Names of output fields. • type: Field type. If type is set to **DATE**, **TIME**, or **TIMESTAMP**, you must specify a time format. If type is set to other values, the time format is invalid. An example time format is **yyyyMMdd HH:mm:ss**. • decimal format: scale and precision of the decimal. • length: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length. • partition key: indicates whether a column is a partition column. You can specify zero or multiple partition columns. If multiple primary keys are configured, they are combined according to the configuration sequence.	map	Yes	None

Data Processing Rule

- The field values are exported to the Hive table.
- If one or more columns are specified as partition columns, the **Partition Handlers** feature is displayed on the **To** page in Step 4 of the job configuration. **Partition Handlers** specifies the number of handlers for processing data partitioning.
- If no column is designated as partition columns, input data does not need to be partitioned, and **Partition Handlers** is hidden by default.

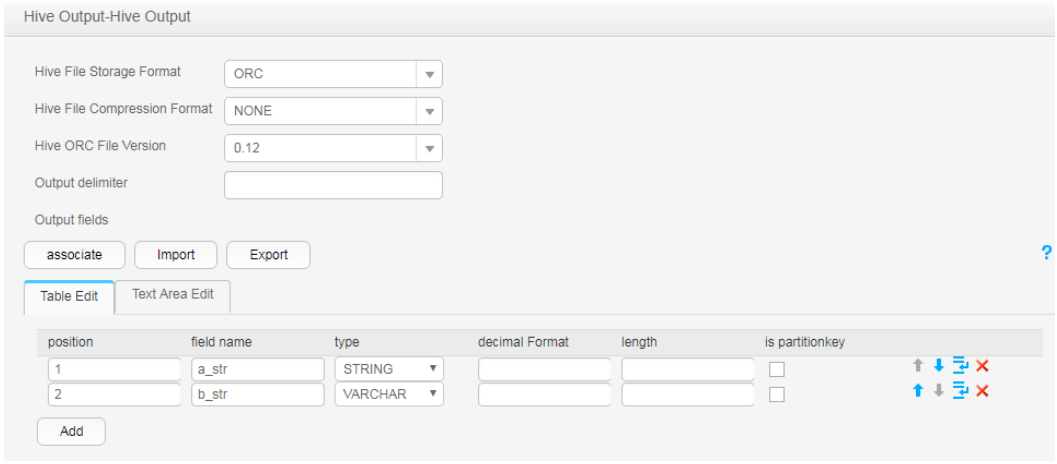
Example

Use the **CSV File Input** operator to generate two fields A and B.

The following figure shows the source file.


```
2016,year
year,2016
```

Configure the **Hive Output** operator to export a_str and b_str to the Hive table.



After the execution is complete, view the table data.

```
0: jdbc:hive2://10.52.0.97:21066/> select * from hive_test;
+------------------+------------------+--+
| hive_test.a_str  | hive_test.b_str  |
+------------------+------------------+--+
| 2016             | year             |
| year             | 2016             |
+------------------+------------------+--+
2 rows selected (1.6 seconds)
```

20.8.4.2 Spark Output

Overview

The **Spark Output** operator exports existing fields to specified columns of a Spark SQL table.

Input and Output

- Input: fields to be exported
- Output: SparkSQL table

Parameter Description

Table 20-119 Operator parameters description

Parameter	Description	Node Type	Mandatory	Default Value
Spark file storage format	SparkSQL configuration file storage format. CSV, ORC, RC and PARQUET are supported at present. **NOTE** • PARQUET is a column-based storage format. In this format, the output field names of Loader be the same as the field names in the SparkSQL table. • For Hive of versions later than 1.2.0, a field name, instead of field number, is used to parse ORC files. Therefore, the output field names of Loader must be the same as those in the SparkSQL table.	enum	Yes	CSV
Spark file compression format	SparkSQL table file compression format. Select a format from the drop-down list. If you select **NONE** or do not set this parameter, data is not compressed.	enum	Yes	NONE
Spark ORC file version	Version of the ORC file (when the storage format of the SparkSQL table file is ORC).	enum	Yes	0.12
Output delimiter	Delimiter.	string	Yes	None

Parameter	Description	Node Type	Mandatory	Default Value
Output fields	Information about output fields: • position: Position of output fields. • field name: Names of output fields. • type: Field type. If type is set to **DATE**, **TIME**, or **TIMESTAMP**, you must specify a time format. If type is set to other values, the time format is invalid. An example time format is **yyyyMMdd HH:mm:ss**. • decimal format: scale and precision of the decimal. • length: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length. • partition key: indicates whether a column is a partition column. You can specify zero or multiple partition columns. If multiple primary keys are configured, they are combined according to the configuration sequence.	map	Yes	None

Data Processing Rule

- The field values are exported to the SparkSQL table.
- If one or more columns are specified as partition columns, the **Partition Handlers** feature is displayed on the **To** page in Step 4 of the job configuration. **Partition Handlers** specifies the number of handlers for processing data partitioning.
- If no column is designated as partition columns, input data does not need to be partitioned, and **Partition Handlers** is hidden by default.

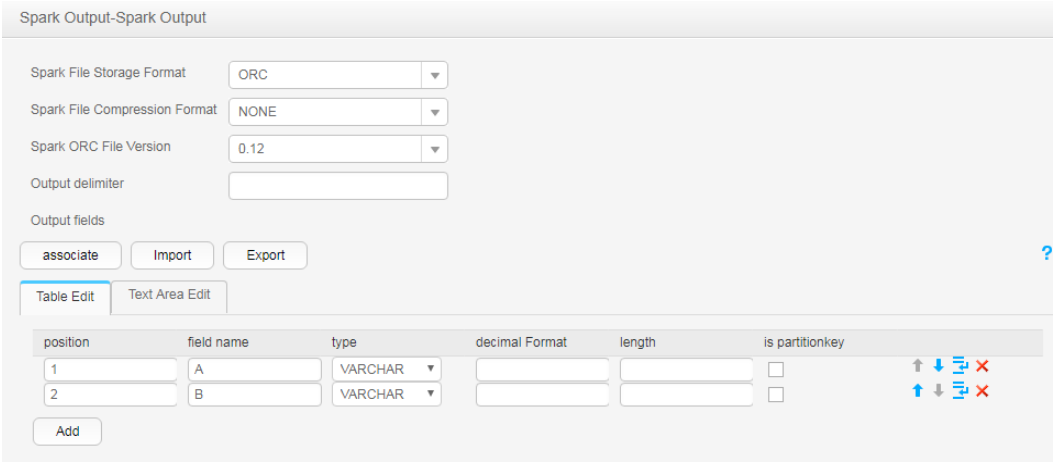
Example

Use the **CSV File Input** operator to generate two fields A and B.

The following figure shows the source file.


```
2016,year
year,2016
```

Configure the **Spark Output** operator to export A and B to the SparkSQL table.



20.8.4.3 Table Output

Overview

The **Table Output** operator exports output fields to specified columns in a relational database table.

Input and Output

- Input: fields to be exported
- Output: relational database table

Parameters

Table 20-120 Operator parameters description

Parameter	Description	Node Type	Mandatory	Default Value
Output delimiter	Delimiter. **NOTE** This configuration applies only to the MySQL dedicated connector. If the data column content contains the default delimiter, you need to set a user-defined delimiter. Otherwise, data disorder may occur.	string	No	,

Parameter	Description	Node Type	Mandatory	Default Value
Line delimiter	Line delimiter, which can be any string specified by users based on the actual situation. Any character string is supported. The OS line delimiter is used by default. **NOTE** This configuration applies only to the MySQL dedicated connector. If the data column content contains the default delimiter, you need to set a user-defined delimiter. Otherwise, data disorder may occur.	string	No	\n
Output fields	Information about relational database output fields: ● field name: Names of output fields. ● table column name: Names of database table columns. ● type: Field type. The value must be consistent with the field type configured in the database. ● length: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length.	map	Yes	None

Data Processing Rule

The field values are exported to the table.

Example

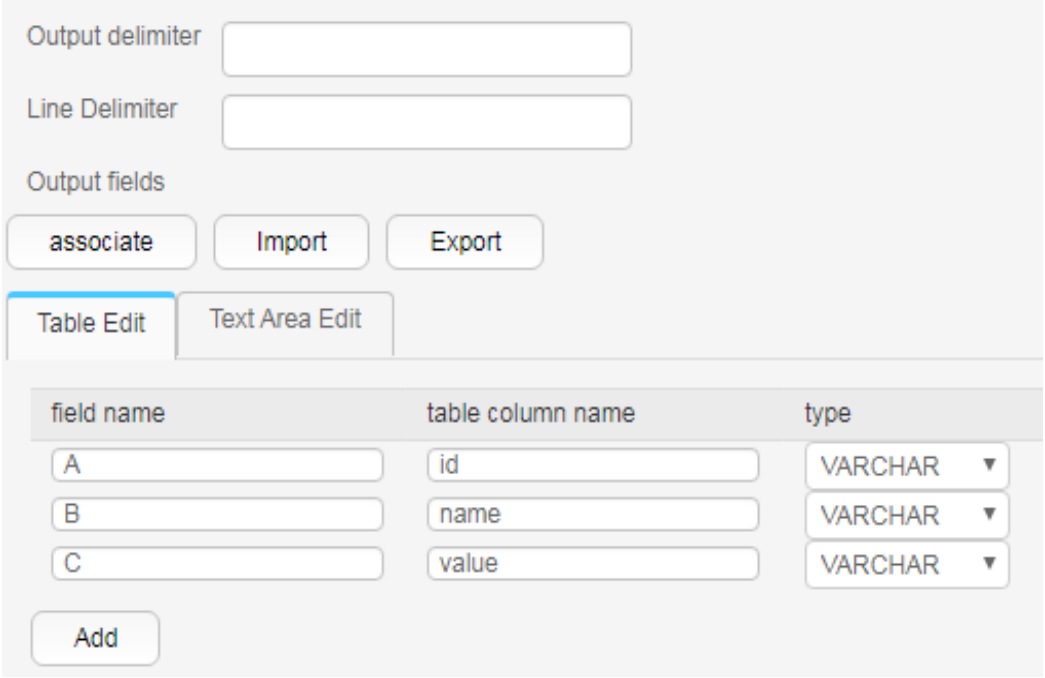
Use the data export from HBase to sqlserver2014 as an example.

In sqlserver2014, run the following statement to create an empty data test_1 for storing HBase data. Run the following statement:

create table test_1 (id int, name text, value text);

Use the HBase Input operator to generated three fields A, B, and C.

Use the **Table Output** operator to export A, B, and C to the test_1 table.



The command output is as follows:

	id	name	value
1	1	zhangshan	zhang
2	2	lisi	li
3	3	wangwu	wang

20.8.4.4 File Output

Overview

The **File Output** operator uses delimiters to concatenate existing fields and exports new fields to a file.

Input and Output

- Input: fields to be exported
- Output: files

Parameter Description

Table 20-121 Operator parameters description

Parameter	Description	Type	Mandatory	Default Value
Output delimiter	Set a delimiter.	string	Yes	None

Parameter	Description	Type	Mandatory	Default Value
Line breaker	Line delimiter, which can be any string specified by users based on the actual situation. Any character string is supported. The OS line delimiter is used by default.	string	No	\n
Output fields	Information about output fields: • **position**: Position of output fields. • **field name**: Names of output fields. • **type**: Field type. If type is set to **DATE**, **TIME**, or **TIMESTAMP**, you must specify a time format. If type is set to other values, the time format is invalid. The example time format is **yyyyMMdd HH:mm:ss**. • **length**: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length.	map	No	None

Data Processing Rule

The field is exported to a file.

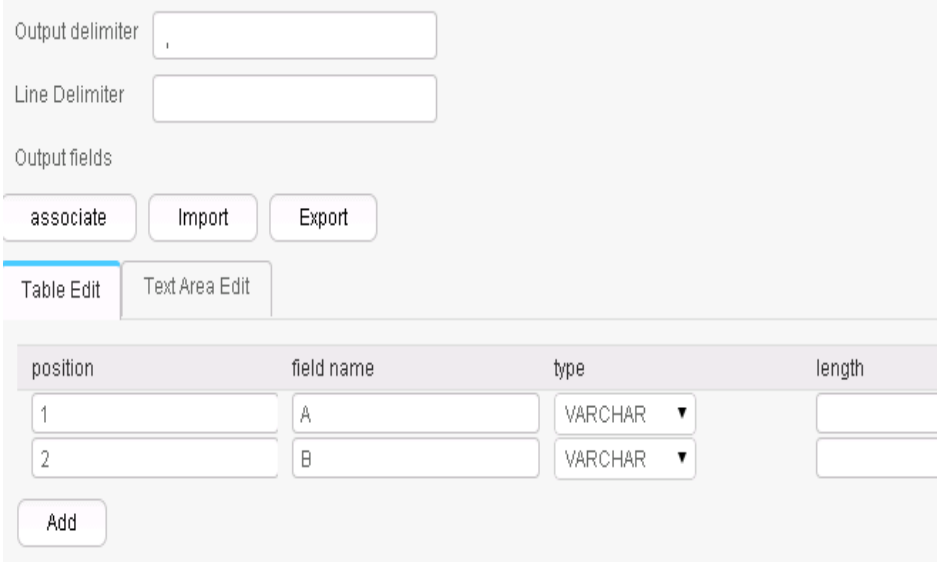
Example

Use the **CSV File Input** operator to generate two fields A and B.

The following figure shows the source file.

```
aaa,product
bbb,Bigdata
```

Configure the **File Output** operator, set **Output delimiter** to a comma (,), and export A and B to a file, as shown in the following figure.



The following figure shows the result.

```
aaa,product
bbb,Bigdata
```

20.8.4.5 HBase Output

Overview

The **HBase Output** operator exports existing fields to specified columns of an HBase Outputtable.

Input and Output

- Input: fields to be exported
- Output: HBase table

Parameters

Table 20-122 Operator parameters description

Parameter	Description	Node Type	Mandatory	Default Value
HBase table type	HBase table type. The options include normal (common HBase table) and phoenix.	enum	Yes	normal

Parameter	Description	Node Type	Mandatory	Default Value
NULL value processing mode	Null value processing mode. Selecting the option button indicates to convert null values to empty strings and save them. Deselecting the option button indicates the data is not saved.	boolean	No	The option button is not selected.
HBase output fields	HBase output information: ● field name: Names of output fields. ● table name: HBase table name. ● family name: HBase column family name. ● column name: HBase column name. ● type: Field type. If type is set to **DATE**, **TIME**, or **TIMESTAMP**, you must specify a time format. If type is set to other values, the time format is invalid. An example time format is **yyyyMMdd HH:mm:ss**. ● length: Field value length. If the actual field value is excessively long, the value is cut based on the configured length. When **type** is set to **CHAR**, spaces are added to the field value for supplement if the actual field value length is less than the configured length. When **type** is set to **VARCHAR**, no space is added to the field value for supplement if the actual field value length is less than the configured length. ● Primary Key: Indicates whether a column is a primary key column. A common HBase table can have only one primary key, while a phoenix table can have multiple primary keys. If multiple primary keys are configured, they are combined according to the configuration sequence. At least one primary key column must be configured.	map	Yes	None

Data Processing Rule

- The field values are exported to the HBase table.
- When the original data contains NULL values, if the **NULL value processing mode** is selected, the NULL values are converted to empty strings and saved. If the **NULL value processing mode** button is not selected, the data is not saved.

Example

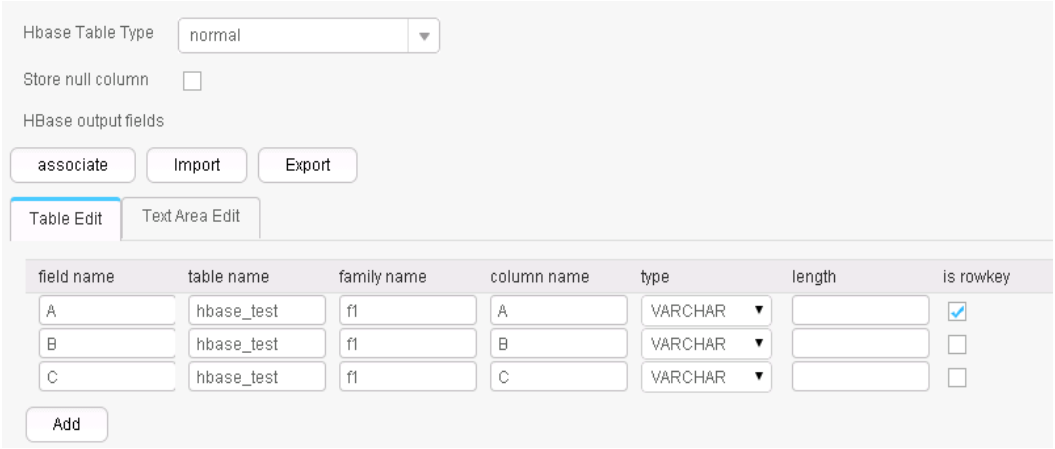
Using table input as an example, after the fields are generated, the HBase Output operator exports them to the related HBase table and stores the data in the test table, as shown in the following figure.

	id	name	value
1	1	zhangshan	zhang
2	2	lisi	li
3	3	wangwu	wang

Create an HBase table.

create 'hbase_test','f1','f2';

Configure the **HBase Output** operator, as shown in the following figure.



After the job execution is complete, view the data in the hbase_test table.

```
hbase(main):001:0> scan 'hbase_test'
ROW                                           COLUMN+CELL
 1                                            column=f1:B, timestamp=1455855645760, value=zhangshan
 1                                            column=f1:C, timestamp=1455855645760, value=zhang
 2                                            column=f1:B, timestamp=1455855645760, value=lisi
 2                                            column=f1:C, timestamp=1455855645760, value=li
 3                                            column=f1:B, timestamp=1455855645760, value=wangwu
 3                                            column=f1:C, timestamp=1455855645760, value=wang
3 row(s) in 0.2720 seconds
```

20.8.4.6 ClickHouse Output

Overview

The **ClickHouse Output** operator exports existing fields to specified columns of a ClickHouse table.

Input and Output

- Input: fields to be exported
- Output: ClickHouse table

Parameters

Table 20-123 Operator parameters

Parameter	Description	Type	Mandatory	Default Value
ClickHouse database name	Database where the ClickHouse table is located.	string	Yes	default
ClickHouse table name	Name of the ClickHouse table to which data is written.	string	Yes	None

Data Processing Rule

The field values are exported to the ClickHouse table.

Example

Use the **CSV File Input** operator to generate 12 fields.

The following figure shows the source file.

```
1,'b','abcd','2021-06-15','12:00:06','2021-06-15 12:00:06',1,12,6.8,18.6,12.8,true
2,'abc','abcd','2021-06-15','12:00:06','2021-06-15 12:00:06',1,12,6.8,18.6,12.8,true
3,'ab','abcd','2021-06-15','12:00:06','2021-06-15 12:00:06',1,12,6.8,18.6,12.8,true
4,'abcdef','abcd','2021-06-15','12:00:06','2021-06-15 12:00:06',1,12,6.8,18.6,12.8,true
5,'a','abcd','2021-06-15','12:00:06','2021-06-15 12:00:06',1,12,6.8,18.6,12.8,true
6,'bg','cde','2020-06-15','13:00:06','2021-06-15 12:00:06',1,12,6.8,18.6,12.8,true
7,'f','cde','2020-06-15','13:00:06','2021-06-15 12:00:06',1,12,6.8,18.6,12.8,true
8,'h','cde','2020-06-15','13:00:06','2021-06-15 12:00:06',1,12,6.8,18.6,12.8,true
```

Run the following statements to create a ClickHouse table:

CREATE TABLE IF NOT EXISTS testck4 ON CLUSTER default_cluster(

a Int32,

b VARCHAR(100) NOT NULL,

c char(100),

d DateTime,

e DateTime,

f DateTime,

g smallint,

h bigint,

l Float32,

j Float64,

k decimal(10,2),

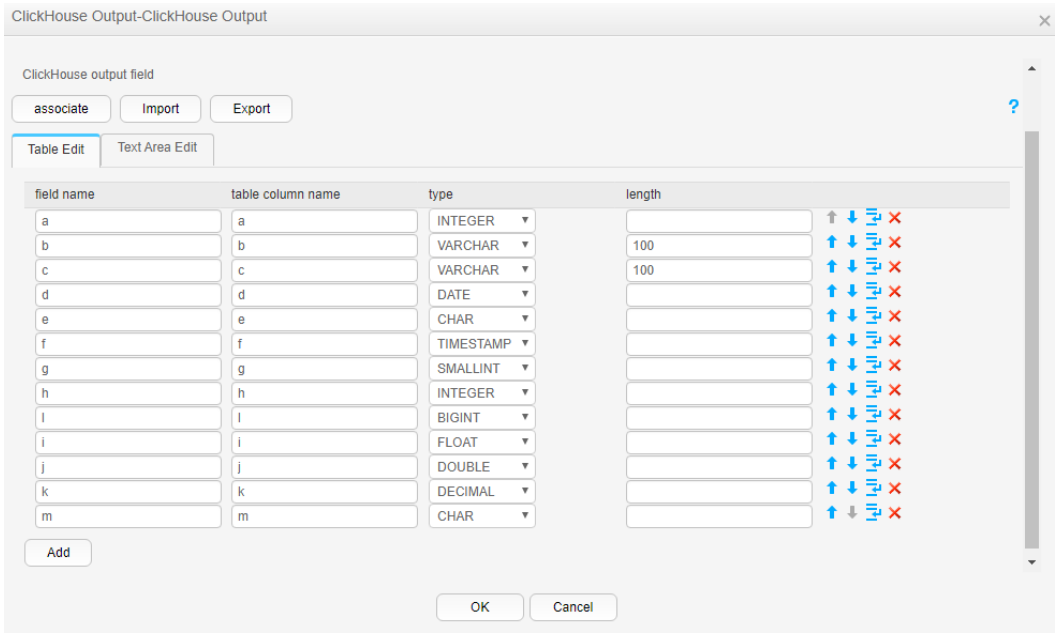
m boolean

)

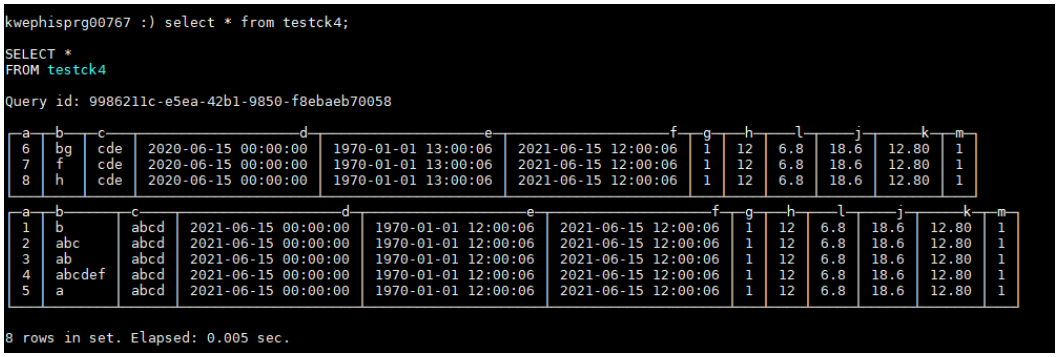
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/testck4', '{replica}')

PARTITION BY toYYYYMM(d)ORDER BY a;

Configure the **ClickHouse Output** operator, as shown in the following figure.



After the job execution is complete, view the data in the **testck4** table.



20.8.5 Managing Loader Operator Configurations

Scenario

This section describes how to associate, import, or export the field configuration information of an operator when creating or editing a Loader job.

- Associating the field configuration of an operator

  Associate the field configuration information of an input operator with an output operator.
- Editing the field configuration of an operator

  Edit the field configuration information of an operator.
- Importing the field configuration of an operator

  Import the field configuration information to an operator by using an operator export file or operator template file.
- Exporting the field configuration of an operator

  Export the field configuration information of an operator to a JSON file and save the file to a local directory.

Prerequisites

You have obtained the username and password for logging in to the Loader web UI.

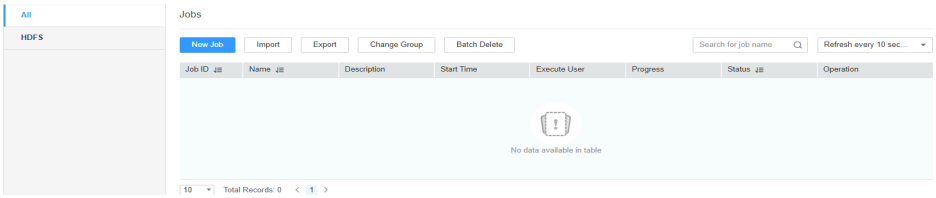
Procedure

- **Associating Field Configuration of an Operator**

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

   Figure 20-87 Loader web UI

Step 2 Edit an existing job or create a new job. The **Transform** page is displayed.

Step 3 Double-click a specified input operator (such as **CSV File Input**) to go to the edit page. Add the configuration information to the parameter table of the input field.

Step 4 Double-click a specified output operator (such as **File Output**) to go to the edit page, click **associate**, and select the required field information in the displayed **associate** dialog box.

NOTE

- The field name already exists in the field table of the output operator and is not displayed in the **associate** window.
- You can also select the required field from the **field name** list. The corresponding configuration information is displayed in the parameter table of the output field.

Step 5 Click **OK**. The selected field is displayed in the parameter table of the output field.

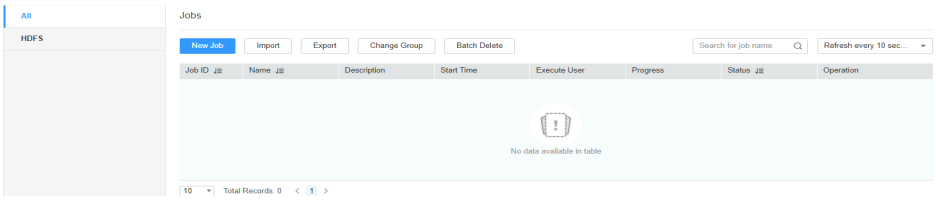
----End

- **Editing the Field Configuration of an Operator**

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-88 Loader web UI

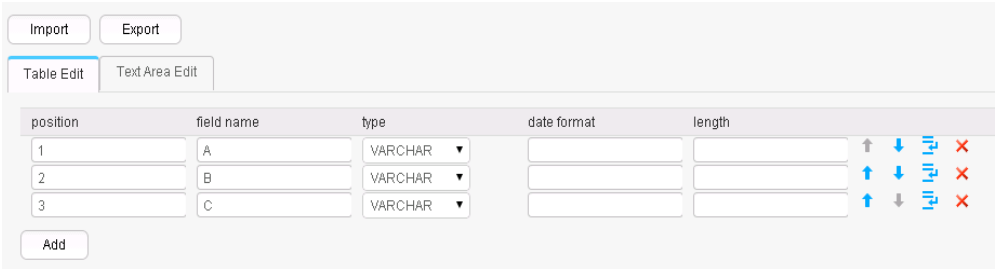


Step 2 Edit an existing job or create a new job. The **Transform** page is displayed.

Step 3 Double-click a specified operator (such as **CSV File Input**) to go to the edit page. On the **Table Edit** tab page of the input field, click **Add** and enter the field information based on the parameter requirements of the operator.

Step 4 You can move (up or down), insert a row under, and delete a field by clicking buttons corresponding to the field.

Click **Text Area Edit** to edit the field list in text format. Use commas (,) to separate field attributes.



Step 5 Click **OK**.

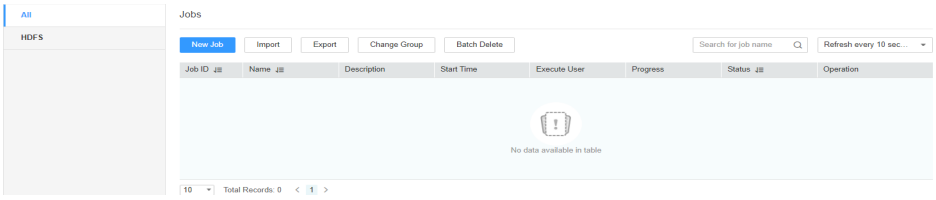
----End

- **Importing the Field Configuration of an Operator**

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-89 Loader web UI



Step 2 Edit an existing job or create a new job. The **Transform** page is displayed.

Step 3 Double-click a specified operator to go to the editing page and add related configuration information to the parameter table of the input or output field. Click **Import**.

Step 4 Select an import type.

- Export File

  Field configuration information is imported by using the JSON file exported by the operator.

- Specified Template

  Field configuration information is imported by using the TXT file compiled based on the operator template.

Step 5 Click  and select the upload file path.

Step 6 Click **Upload**. The field configuration information is imported to the operator.

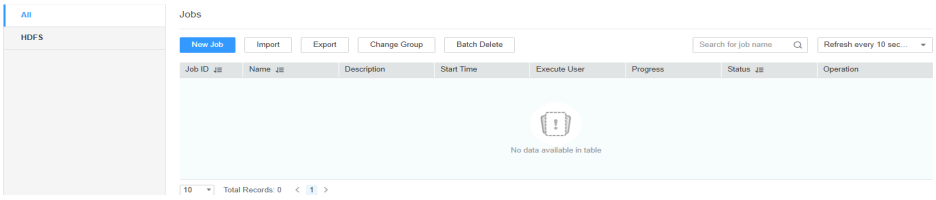
----End

- **Exporting the Field Configuration of an Operator**

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
2. Choose **Cluster** > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 20-90 Loader web UI



Step 2 Edit an existing job or create a new job. The **Transform** page is displayed.

Step 3 Double-click a specified operator to go to the editing page, add related configuration information to the parameter table of the input or output field, and click **Export**.

Step 4 Select an export type.

- All

  All field information is exported as a JSON file and saved to a local directory.

- Specified Field Name

  Fields selected in the field list are exported as a JSON file and saved to a local directory.

Step 5 Click **OK**.

----End

20.8.6 Using Macro Definitions in Configuration Items

When creating or editing Loader jobs, users can use macro definitions during parameter configuration. Then the parameters can be automatically changed to corresponding macro values when a job is implemented.

NOTE

- The macro definitions take effect in the job only.
- Macro definitions can be imported and exported together with an import or export job. If a job uses macro definitions, the exported job includes the macro definitions. Macro definitions are imported by default when a job is imported.
- For details about the format of the first parameter in the **dateformat**time macro, see **java.text.SimpleDateFormat.java**. The restrictions of the target system must be followed. For example, HDFSor OBS directories do not support special characters.

Macro Definitions of Loader

At present, Loader supports the following time macro definitions by default:

Table 20-124 Common macro definitions of Loader

Name	Result After the Replacement	Description
@{dateformat("yyyy-MM-dd")}@	2016-05-17	Indicates the current date.
@{dateformat("yyyy-MM-dd HH:mm:ss")}@	2016-05-17 16:50:00	Indicates current date and time
@{timestamp()}@	1463476137557	Indicates milliseconds since 1970.

Name	Result After the Replacement	Description
@{dateformat("yyyy-MM-dd HH:mm:ss",-7,DAYS)}@	2016-05-10 16:50:00	Indicates the latest seven days (the present time minus seven days). The second parameter supports addition and subtraction. The third parameter is a time unit for calculation. According to definitions in the **java.util.concurrent.TimeUnit.java**, time units include DAYS, HOURS, MINUTES, and SECONDS.

In the following scenarios, parameters can be configured by using macro definitions.

- Specifying a data directory that is named by the current date

  The parameter is set to **/user/data/inputdate_@{dateformat("yyyy-MM-dd")} @**.
- Querying data in the latest seven days by using SQL

  ```
  select * from table where time between '@{dateformat("yyyy-MM-dd HH:mm:ss",-7,DAYS)}@' and '@{dateformat("yyyy-MM-dd HH:mm:ss")}@'
  ```
- Specifying a table that is named by the current date

  The parameter is set to **table_@{dateformat("yyyy-MM-dd")} @parmvalue**.

20.8.7 Operator Data Processing Rules

In Loader data import and export tasks, each operator defines different processing rules for null values and empty strings in raw data. Dirty data cannot be imported or exported.

The following table describes the operator data processing rules for each conversion procedure.

Table 20-125 Data processing rules

Procedure	Description
CSV file input	• If a delimiter appears twice consecutively in the original data, an empty string field is generated. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data. • If a type conversion error occurs, the current data is saved as dirty data.
Fixed file input	• If the original data includes null values, no conversion is performed. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data. • If the configured field conversion type is different from the actual type of the original data, all data becomes dirty data. For example, convert the string type to the numeric type. • If the configured field split length is greater than the length of the original field value, the data split fails and the current line becomes dirty data.
Table input	• If the original data includes null values, no conversion is performed. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data. • If the configured field conversion type is different from the actual type of the original data, all data becomes dirty data. For example, convert the string type to the numeric type.

Procedure	Description
HBase input	• If the original data includes null values, no conversion is performed. • If the HBase table name is incorrect, all data becomes dirty data. • If the primary key column is not configured in **Is rowkey**, all data becomes dirty data. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data. • If the configured field conversion type is different from the actual type of the original data, all data becomes dirty data. For example, convert the string type to the numeric type.
Long integer time conversion	• If the original data includes null values, no conversion is performed. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data. • If a type conversion error occurs, the current data is saved as dirty data.
Null value conversion	• If the original data contains null values, data is converted to a specified value. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data.
Random value conversion	Processing of null value and empty string is not involved, and dirty data is not generated.
Constant field addition	Processing of null value and empty string is not involved, and dirty data is not generated.
Concat fields	• If the original data contains null values, data is converted to empty string. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data.
Extracts fields	• If the original data contains null values, the current line becomes dirty data. • If the number of field columns after separation is greater than the actual number allowed by the original data, the line will become dirty data.

Procedure	Description
Modulo integer	• If the original data contains null values, the current line becomes dirty data. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data. • If data type conversion fails, the current line becomes dirty data.
String cut	• If the input data is null, no conversion is performed. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data. • If the start or end position of the string to be truncated is greater than the length of the input field, the current line becomes dirty data.
EL operation	• If the input data is null, no conversion is performed. • Enter the value of one or more fields and output the calculation result. • When the input type is incompatible with the operator, the current row is dirty data.
String case conversion	• If the input data is null, no conversion is performed. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data.
String reverse	• If the input data is null, no conversion is performed. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data.
String trim	• If the input data is null, no conversion is performed. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data.

Procedure	Description
Filter rows	• When the condition logic is **AND**, if no filter condition is added, all data becomes dirty data; if the original data meets all the added filter conditions, the current line becomes dirty data. • When the condition logic is **OR**, if no filter condition is added, all data becomes dirty data; if the original data meets all the added filter conditions, the current line becomes dirty data.
File output Table output	• If the input data is null, no conversion is performed. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data. • If data type conversion fails, the current line becomes dirty data.
HBase output	• If the original data contains null values and **Store null column** is set to **true**, data is converted to empty string and saved. If **Store null column** is set to **false**, data will not be saved. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data. • If data type conversion fails, the current line becomes dirty data.
Hive output	• If one or more columns are designated as partition columns, the **Partition Handlers** feature is displayed on the **To** page. **Partition Handlers** specifies the number of handlers for processing data partitioning. • If no column is designated as partition columns, input data does not need to be partitioned, and **Partition Handlers** is hidden by default. • It can be configured that all data becomes dirty data when the number of input field columns is greater than the number of field columns actually included in the original data. • If data type conversion fails, the current line becomes dirty data.

20.9 Loader Client Tools

20.9.1 Running a Loader Job by Using Commands

Scenario

Generally, users can manually manage data import and export jobs on the Loader UI. If you need to update and run Loader jobs by executing the shell script, you must configure the installed Loader client.

NOTE

Loader is incompatible with the client of an earlier version. If you reinstall the cluster or the Loader service, download and install the client again, and then use the client.

Prerequisites

- The Loader client has been installed. During the installation of the Loader client using a non-root user, if another user wants to use the client, the user needs to be authorized by the user who installs the client or a user with more rights (the Loader client installation directory needs to be granted with right 755). Please pay attention to the security problems after the authorization.
- The user for accessing the Loader service has been created. If the user is a machine-machine user, the keytab file must be downloaded..

Procedure

Step 1 Configure the Loader shell client.

1. Log in to the node where the client is located as the user who installs the client.
2. Run the following command to disable logout upon timeout:

   TMOUT=0

   NOTE

   After the operations in this section are complete, run the **TMOUT=***Timeout interval* command to restore the timeout interval in a timely manner. For example, **TMOUT=600** indicates that a user is logged out if the user does not perform any operation within 600 seconds.
3. Run the following command to go to the Loader client installation directory, for example, **/opt/client/Loader**:

   cd /opt/client/Loader
4. Run the following command to configure environment variables:

   source/opt/client/bigdata_env
5. If the cluster is in security mode, run the following command to authenticate the user. In normal mode, user authentication is not required.

   kinit *Component service user*
6. Run the following command to modify the tool authorization configuration file **login-info.xml**, save the file, and exit. For the parameters in the configuration file, see **Table 20-126**.

   vi loader-tools-1.99.3/loader-tool/job-config/login-info.xml

Table 20-126 Parameters of **login-info.xml**

Parameter	Description
hadoop.config.path	Storage directory of the **core-site.xml**, **hdfs-site.xml**, and **krb5.conf** configuration files of the MRS cluster. These three files are stored in the ***Loader Client installation directory*/Loader/loader-tools-1.99.3/loader-tool/hadoop-config/** directory by default.
authentication.type	Authentication type of the Loader service. Set this parameter based on MRS cluster authentication mode. – **kerberos** indicates the security mode. – **simple** indicates the normal mode.
user.keytab	Whether to use the keytab file for authentication. The options are **true**, and **false**.
authentication.user	User for login when the normal mode or password authentication is used. In the keytab login mode, this parameter does not need to be set.
authentication.password	Encrypted password of the user for accessing the Loader service if the keytab file authentication is not used in the security mode. **NOTE** Run the following command to encrypt the password as the user who installs the client. When the encryption tool runs for the first time, a random dynamic key is automatically generated and stored in **.loader-tools.key**. The encryption tool uses this dynamic key to encrypt passwords every time. After **.loader-tools.key** is deleted, a new random key will be generated and stored in **.loader-tools.key** when the encryption tool runs. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage. **sh** *Loader client installation directory*/**Loader/loader-tools-1.99.3/encrypt_tool** *password*

Parameter	Description
authentication.principal	**Machine-Machine** username for accessing the Loader service when the keytab file authentication is used in the security mode.
authentication.keytab	Absolute keytab file directory of the Machine-Machine user for accessing the Loader service when the keytab file authentication is used in the security mode.
zookeeper.quorum	IP address and port for accessing ZooKeeper. The value format is **IP1:port,IP2:port,IP3:port**. The default port number is 2181.
sqoop.server.list	Floating IP address and port for accessing Loader. The value format is **floatip:port**. The default port number is **21351**.

Step 2 Use the Loader shell client.

1. Run the following command to go to the Loader shell client directory. For example, if the Loader client installation directory is **/opt/client/Loader**, run the following command:

   cd /opt/client/Loader/loader-tools-1.99.3/shell-client/

2. Run the following command to use the Loader shell client to run a job:

   ./submit_job.sh -n *<arg>* **-u** *<arg>* **-jobType** *<arg>* **-connectorType** *<arg>* **-frameworkType** *<arg>*

   Table 20-127 Parameters of the Loader shell client tool

Parameter	Description
-n	(Mandatory) Job name.
-u	(Mandatory) If the parameter is set to **y**, the job parameters are updated and the job is executed. In this scenario, parameters **-jobType**, **-connectorType**, and **-frameworkType** need to be set. If the parameter is set to **n**, the job is directly executed without updating parameters.

Parameter	Description
-jobType	Job type. This parameter is mandatory when **-u** is set to **y**. **import** indicates the data import job. **export** indicates the data export job.
-connectorType	Connector type. This parameter is mandatory when **-u** is set to **y**. Parameters of external data sources can be modified as required. **sftp** indicates the connector is an SFTP connector. – In a data import job, you can modify the source file input path **-inputPath**, the source file encode format **-encodeType**, and the suffix **-suffixName** added to the input file after the source file is imported. – In a data export job, you can modify the output path **-outputPath** or the name of the exported file. **rdb** indicates the connector is a relational database connector. – In a data import job, you can modify the database mode name **-schemaName**, table name **-tableName**, SQL statement **-sql**, names of columns to be imported **-columns**, and names of partition columns **-partitionColumn**. – In a data export job, you can modify the database mode name **-schemaName**, table name **-tableName**, and the temporary table name **-stageTableName**.

Parameter	Description
-frameworkType	Data storage type on MRS. This parameter is mandatory when **-u** is set to **y**. Parameters of data storage types can be modified as required. **hdfs** indicates that the HDFS is used to store data on Hadoop. – In a data import job, you can modify the number of started maps **-extractors** and the storage directory of imported data in the HDFS **-outputDirectory**. – In a data export job, you can modify the number of started maps **-extractors**, the input path of data exported from the HDFS **-inputDirectory**, and the file filter criteria of the data export job **-fileFilter**. **hbase** indicates that HBase is used to store data on MRS. In the data import and export job, you can modify the number of started maps **-extractors**.

----End

Task Examples

- Run a job whose name is **sftp-hdfs** without updating job parameters:

  ./submit_job.sh -n sftp-hdfs -u n

- Update the input path, encoding type, suffix, output path, and number of started maps of the data import job whose name is **sftp-hdfs**, and run the job:

  ./submit_job.sh -n sftp-hdfs -u y -jobType import -connectorType sftp -inputPath /opt/tempfile/1 -encodeType UTF-8 -suffixName '' -frameworkType hdfs -outputDirectory /user/user1/tttest -extractors 10

- Update the database mode, table name, and output path of the data import job whose name is **db-hdfs**, and run the job.

  ./submit_job.sh -n db-hdfs -u y -jobType import -connectorType rdb -schemaName public -tableName sq_submission -sql '' -partitionColumn sqs_id -frameworkType hdfs -outputDirectory /user/user1/dbdbt

20.9.2 loader-tool Usage Guide

Overview

loader-tool is a Loader client tool. It consists of three tools: **lt-ucc**, **lt-ucj**, **lt-ctl**.

Loader supports two modes, parameter mode and job template mode. Either mode can be used to create, update, query, and delete connectors, and to create, update, query, delete, start, and stop Loader jobs.

NOTE

loader-tool implements an asynchronous interface. After a command is submitted, the command output is not returned to the console in real time. Therefore, the results of the creation, update, query, and deletion operations on a connector and the creation, update, query, deletion, start, and stop operations on a Loader job must be confirmed on the Loader WebUI or by querying server logs.

- Parameter mode:

  Add a parameter invoking script with specific parameters.

- Job template mode:

  Change the values of all parameters in a job template and reference the job template when invoking a script.

  After a Loader client is installed, the system automatically generates job templates for various scenarios in the *Loader client installation directory*/**loader-tools-1.99.3/loader-tool/job-config/** directory. The parameters vary according to job templates. Job templates contain information about jobs and associated connectors.

  Job templates are XML files. The file name format is *original data location*-**to**-*new data location*.**xml**, for example, **sftp-to-hdfs.xml**. If a job supports conversion step, a json conversion step configuration file with the same name exists, for example, **sftp-to-hdfs.json**.

  NOTE

  Job templates contain the configuration information of connectors. During the connector creation and updating, only the connector information in job templates is invoked.

Scenarios

The parameters vary according to connectors or jobs.

- To modify some parameters, use the parameter mode.
- To create a connector or job, use the job template mode.

  NOTE

  This tool currently supports the FTP, HDFS, JDBC, MySQL, Oracle, and Oracle dedicated connectors. If other types of connectors are used, you are advised to use the open-source sqoop-shell tool.

Parameters

For example, the Loader client installation directory is **/opt/client/Loader/**.

- **lt-ucc usage description**

  lt-ucc is a connector configuration tool of loader-tool user-configuration-connection and is used to create, update, and delete connectors.

Table 20-128 lt-ucc script parameter description

Parameter	Description	Example Value
-help	Help information.	-
-a <arg>	Connector action. The values include **create**, **update** and **delete** for creating, updating, and deleting connectors respectively.	create
-at <arg>	Login authentication type. The values include **kerberos** and **simple**.	kerberos
-uk <arg>	Whether to use the keytab file.	true
-au <arg>	Login authentication username.	bar
-ap <arg>	Login authentication password. The value must be an encrypted password. The password encryption method is described as follows: **sh** *Loader client installation directory*/**Loader/loader-tools-1.99.3/encrypt_tool** *non-encrypted user password* **NOTE** If a non-encrypted password contains special characters, the special characters must be escaped. For example, the dollar sign ($) is a special character and can be escaped using single quotation marks ('). If a non-encrypted password contains single quotation marks, use double quotation marks to escape the single quotation marks. If a non-encrypted password contains double quotation marks, use backslashes (\) to escape the double quotation marks. For details, see the shell escape character rules.	-
-c <arg>	Login authentication principal.	bar
-k <arg>	Login authentication keytab file.	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config/user.keytab
-h <arg>	Specifies the configuration file path of the MRS cluster.	-h /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config

Parameter	Description	Example Value
-l <arg>	Login template file.	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml
-s <arg>	Floating IP address and port for Loader. Format: *floating IP address: port* The default port is **21351**.	127.0.0.1:21351
-w <arg>	Job template file path for obtaining job details.	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml
-z <arg>	IP address and port number of ZooKeeper quorum instances. The format is *IP address: port*. Use **commas (,)** to separate multiple addresses and port numbers.	127.0.0.0:2181,127.0.0.1:2181
-n <arg>	Connector name	vt_sftp_test
-t <arg>	Connector type	sftp-connector
-P <arg>	Used to update the value of an attribute. The format is -Pparam1=value1. param1 indicates the attribute name of the connector in the job template. Password parameters are required for updating SFTP and FTP connector information. -Pconnection.sftpPassword=*Encrypted password*	-Pconnection.sftpServerIp=10.6.26.11

A complete example is as follows:

./bin/lt-ucc -l /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -n vt_sftp_test -t sftp-connector -Pconnection.sftpPassword=*Password ciphertext* **-Pconnection.sftpServerIp=10.6.26.111 -a update**

Configuration description of a lt-ucc script job template:

Use the operation of saving SFTP data to HDFS as an example. Edit the **sftp-to-hdfs.xml** file in *Loader client installation directory***/loader-tools-1.99.3/loader-tool/job-config/** directory. The connector configuration is as follows:

```
<!-- Database connection information -->
<sqoop.connection name="vt_sftp_test" type="sftp-connector">
<connection.sftpServerIp>10.96.26.111</connection.sftpServerIp>
<connection.sftpServerPort>22</connection.sftpServerPort>
```



```
<connection.sftpUser>root</connection.sftpUser>
<connection.sftpPassword>Encrypted password</connection.sftpPassword>
</sqoop.connection>
```

  - Creation command:

    ./lt-ucc -l /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -w /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/ftp-to-hdfs.xml -a create

  - Update command:

    ./lt-ucc -l /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -w /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/ftp-to-hdfs.xml -a update

  - Deletion command:

    ./lt-ucc -l /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -w /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/ftp-to-hdfs.xml -a delete

- **lt-ucj usage description**

  lt-ucj is a job configuration tool of loader-tool user-configuration-job and is used to create, update, and delete jobs.

  Table 20-129 lt-ucj script parameter description

Parameter	Description	Example Value
-help	Help information.	-
-a <arg>	Job action. The values include **create**, **update**, and **delete** for creating, updating and deleting jobs respectively.	create
-at <arg>	Login authentication type. The values include **kerberos** and **simple**.	kerberos
-uk <arg>	Whether to use the keytab file.	true
-au <arg>	Login authentication username.	bar

Parameter	Description	Example Value
-ap <arg>	Login authentication password. The value must be an encrypted password. The password encryption method is described as follows: **sh** *Loader client installation directory*/**Loader/loader-tools-1.99.3/encrypt_tool** *non-encrypted user password* **NOTE** If a non-encrypted password contains special characters, the special characters must be escaped. For example, the dollar sign ($) is a special character and can be escaped using single quotation marks ('). If a non-encrypted password contains single quotation marks, use double quotation marks to escape the single quotation marks. If a non-encrypted password contains double quotation marks, use backslashes (\) to escape the double quotation marks. For details, see the shell escape character rules.	-
-c <arg>	Login authentication principal.	bar
-k <arg>	Login authentication keytab file.	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config/user.keytab
-h <arg>	Specifies the configuration file path of the MRS cluster.	-h /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config
-l <arg>	Login template file.	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml
-s <arg>	Floating IP address and port for Loader. Format: *floating IP address: port* The default port is 21351.	127.0.0.1:21351

Parameter	Description	Example Value
-w <arg>	Job template file for obtaining job details.	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml
-z <arg>	IP address and port number of ZooKeeper quorum instances. The format is *IP address: port*. Use **commas (,)** to separate multiple addresses and port numbers.	127.0.0.0:2181,127.0.0.1:2181
-n <arg>	Name of the job.	Sftp.to.Hdfs
-cn <arg>	Connector name	vt_sftp_test
-ct <arg>	Connector type	sftp-connector
-t <arg>	Job type. The values include **IMPORT** and **EXPORT**.	IMPORT
-trans <arg>	Job associated conversion step file.	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.json
-priority <arg>	Job priority. The values include **LOW**, **NORMAL**, and **HIGH**.	NORMAL
-queue <arg>	Queues	default
-storageType <arg>	Storage type	HDFS
-P <arg>	Used to update the value of an attribute. The format is -Pparam1=value1. param1 indicates the attribute name of the connector in the job template. Password parameters are required for updating SFTP and FTP connector information. -Pconnection.sftpPassword=*Encrypted password*	-Pconnection.sftpServerIp=10.6.26.11

A complete example is as follows:

./bin/lt-ucj -l /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -n Sftp.to.Hdfs -t IMPORT -ct sftp-connector -Poutput.outputDirectory=/user/loader/sftp-to-hdfs-test8888 -a update

Configuration description of a lt-ucj script job template:

Use the operation of saving SFTP data to HDFS as an example. Edit the file *loader client installation directory***/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml**. The job configuration is as follows:

```
<!--Job name, globally unique.-->
<sqoop.job name="Sftp.to.Hdfs" type="IMPORT" queue="default" priority=" Priority NORMAL ">

<!-- External data source parameter configuration -->
<data.source connectionName="vt_sftp_test" connectionType="sftp-connector">
<file.inputPath>/opt/houjt/hive/all</file.inputPath>
<file.splitType>FILE</file.splitType>
<file.filterType>WILDCARD</file.filterType>
<file.pathFilter>*</file.pathFilter>
<file.fileFilter>*</file.fileFilter>
<file.encodeType>GBK</file.encodeType>
<file.suffixName></file.suffixName>
<file.isCompressive>FALSE</file.isCompressive>
</data.source>

<!-- MRS cluster, parameter configuration -->
<hadoop.source storageType="HDFS" >
<output.outputDirectory>/user/loader/sftp-to-hdfs</output.outputDirectory>
<output.fileOprType>OVERRIDE</output.fileOprType>
<throttling.extractors>3</throttling.extractors>
<output.fileType>TEXT_FILE</output.fileType>
</hadoop.source>

<!-- Job associated conversion step file -->
<sqoop.job.trans.file>/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.json</sqoop.job.trans.file>
</sqoop.job>
```

- Creation command:

  ./bin/lt-ucj -l /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -w /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml -a create

- Update command:

  ./bin/lt-ucj -l /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -w /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml -a update

- Deletion command:

  ./bin/lt-ucj -l /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -w /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml -a delete

- **lt-ctl usage description**

  lt-ctl is a job management tool of loader-tool controller and is used to start or stop jobs, query job status and progress, and check whether jobs are running.

  Table 20-130 lt-ctl script parameter description

Parameter	Description	Example Value
-help	Help information.	-

Parameter	Description	Example Value
-a <arg>	Job action. The values include **status**, **start**, **stop**, and **is running** for querying job status, starting or stopping jobs, and checking whether jobs are running.	create
-at <arg>	Login authentication type. The values include **kerberos** and **simple**.	kerberos
-uk <arg>	Whether to use the keytab file.	true
-au <arg>	Login authentication username.	bar
-ap <arg>	Login authentication password. The value must be an encrypted password. The password encryption method is described as follows: **sh** *Loader client installation directory*/**Loader/loader-tools-1.99.3/encrypt_tool** *non-encrypted user password* **NOTE** If a non-encrypted password contains special characters, the special characters must be escaped. For example, the dollar sign ($) is a special character and can be escaped using single quotation marks ('). If a non-encrypted password contains single quotation marks, use double quotation marks to escape the single quotation marks. If a non-encrypted password contains double quotation marks, use backslashes (\) to escape the double quotation marks. For details, see the shell escape character rules.	-
-c <arg>	Login authentication principal.	bar
-k <arg>	Login authentication keytab file.	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config/user.keytab
-h <arg>	Specifies the configuration file path of the MRS cluster.	-h /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config

Parameter	Description	Example Value
-l <arg>	Login template file.	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml
-n <arg>	Name of the job.	Sftp.to.Hdfs
-s <arg>	Floating IP address and port for Loader. Format: *floating IP address: port* The default port is 21351.	127.0.0.1:21351
-w <arg>	Job template file for obtaining job details.	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml
-z <arg>	IP address and port number of ZooKeeper quorum instances. The format is *IP address: port*. Use **commas (,)** to separate multiple addresses and port numbers.	127.0.0.0:2181,127.0.0.1:2181

- Command for starting jobs:

  ./bin/lt-ctl -l /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -n Sftp.to.Hdfs -a start
- Command for viewing job status:

  ./bin/lt-ctl -l /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -n Sftp.to.Hdfs -a status
- Command for checking whether jobs are running:

  ./bin/lt-ctl -l /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -n Sftp.to.Hdfs -a isrunning
- Command for stopping jobs:

  ./bin/lt-ctl -l /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -n Sftp.to.Hdfs -a stop

20.9.3 loader-tool Usage Example

Scenario

loader-tool can be used to create, update, query, and delete a connector or job by using a job template or setting parameters.

This section describes how to use loader-tool in the job template mode. The job of importing data from the SFTP server to HDFS is used as an example.

Prerequisites

The Loader client has been installed and configured. For details, see **Running a Loader Job by Using Commands**.

Procedure

Step 1 Log in to the node where the client is located as the user who installs the client.

Step 2 Run the following command to go to the loader-tool directory on the Loader client, for example, **/opt/client/Loader/**:

cd /opt/client/Loader/loader-tools-1.99.3/loader-tool/

Step 3 Run the following command to modify the existing job template. For example, if the job template **sftp-to-hdfs.xml** already exists in the **/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/**, run the following command:

vi /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml

```
<root>
<!-- Database connection information -->
<sqoop.connection name="vt_sftp_test" type="sftp-connector">
<connection.sftpServerIp>10.96.26.111</connection.sftpServerIp>
<connection.sftpServerPort>22</connection.sftpServerPort>
<connection.sftpUser>root</connection.sftpUser>
<connection.sftpPassword>Encrypted password</connection.sftpPassword>
</sqoop.connection>

<!--Job name, globally unique.-->
<sqoop.job name="Sftp.to.Hdfs" type="IMPORT" queue="default" priority="NORMAL">
<data.source connectionName="vt_sftp_test" connectionType="sftp-connector">
<file.inputPath>/opt/houjt/hive/all</file.inputPath>
<file.splitType>FILE</file.splitType>
<file.filterType>WILDCARD</file.filterType>
<file.pathFilter>*</file.pathFilter>
<file.fileFilter>*</file.fileFilter>
<file.encodeType>GBK</file.encodeType>
<file.suffixName></file.suffixName>
<file.isCompressive>FALSE</file.isCompressive>
</data.source>

<hadoop.source storageType="HDFS" >
<output.outputDirectory>/user/loader/sftp-to-hdfs</output.outputDirectory>
<output.fileOprType>OVERRIDE</output.fileOprType>
<throttling.extractors>3</throttling.extractors>
<output.fileType>TEXT_FILE</output.fileType>
</hadoop.source>

<sqoop.job.trans.file></sqoop.job.trans.file>
</sqoop.job>
</root>
```


NOTE

Each Loader job needs to be associated with a connector. Connectors are used to read data from external data sources when data is imported to a cluster and used to write data into external data sources when data is exported from the cluster. In the preceding example, an SFTP data source connector is configured. To configure an SFTP and FTP data source connector, a password needs to be set and encrypted. The password encryption method is described as follows:

1. Run the following command to go to the **loader-tools-1.99.3** directory. For example, if the Loader client installation directory is **/opt/hadoopclient/Loader**, run the following command:

   cd /opt/hadoopclient/Loader/loader-tools-1.99.3

2. Run the following command to encrypt the non-encrypted password:

   ./encrypt_tool *Unencrypted password*

Step 4 Run the following command to go to the directory where loader-tool is located:

cd /opt/client/Loader/loader-tools-1.99.3/loader-tool

Step 5 Run the following command to use the lt-ucc tool to create a connector:

./bin/lt-ucc -l /opt/client/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -w /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml -a create

If no error is reported and the following information is displayed, the connector creation task is submitted successfully:

```
User login success. begin to execute task.
```

Step 6 Run the following command to use the lt-ucj tool to create a job:

./bin/lt-ucj -l /opt/client/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -w /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml -a create

If no error is reported and the following information is displayed, the job creation task is submitted successfully:

```
User login success. begin to execute task.
```

Step 7 Run the following command to use the lt-ctl tool to submit the job:

./bin/lt-ctl -l /opt/client/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -n Sftp.to.Hdfs -a start

If the following information is displayed, the job is submitted successfully:

```
Start job success.
```

Step 8 Run the following command to view the job status:

./bin/lt-ctl -l /opt/client/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml -n Sftp.to.Hdfs -a status

```
Job:Sftp.to.Hdfs
Status:RUNNING
Progress: 0.0
```

----End

20.9.4 schedule-tool Usage Guide

Overview

schedule-tool is used to submit jobs of SFTP data sources. You can modify the input path and file filtering criteria before submitting a job. You can modify the output path if the target source is HDFS.

Parameters

Table 20-131 Configuration parameters of schedule.properties

Configuration parameters	Description	Example Value
server.url	Floating IP address and port for Loader. The default port is 21351. For compatibility, multiple IP addresses and ports can be configured and need to be separated by commas (,). The first IP address and port must be those of Loader. The others can be configured based on service requirements.	10.96.26.111:21351,127.0.0.2:21351
authentication.type	Login authentication mode. • **kerberos** indicates that the security mode is used and Kerberos authentication is performed. Kerberos authentication provides two authentication modes: the password mode and the keytab file mode. • **simple** indicates that the normal mode is used and Kerberos authentication is not performed.	kerberos
authentication.user	User for login when the normal mode or password authentication is used. In the keytab login mode, this parameter does not need to be set.	bar

Configuration parameters	Description	Example Value
authentication.password	User password for login when the password authentication mode is used. In the normal mode or keytab login mode, this parameter does not need to be set. The password needs to be encrypted. The encryption method is described as follows: 1. Go to the directory where **encrypt_tool** is located. For example, if the Loader client installation directory is **/opt/hadoopclient/Loader**, run the following command: **cd /opt/hadoopclient/Loader/loader-tools-1.99.3** 2. Run the following command to encrypt the non-encrypted password. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage. **./encrypt_tool** *Unencrypted password* The obtained encrypted password is used as the value of **authentication.password**. **NOTE** If a non-encrypted password contains special characters, the special characters must be escaped. For example, the dollar sign ($) is a special character and can be escaped using single quotation marks ('). If a non-encrypted password contains single quotation marks, use double quotation marks to escape the single quotation marks. If a non-encrypted password contains double quotation marks, use backslashes (\) to escape the double quotation marks. For details, see the shell escape character rules.	-

Configuration parameters	Description	Example Value
use.keytab	Whether to use the keytab mode to log in. • **true** indicates using the keytab file to log in. • **false** indicates using the password to log in.	true
client.principal	User principal for accessing the Loader service when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	loader/hadoop.*System domain name* **NOTE** You can log in to FusionInsight Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and view the value of **Local Domain**, which is the current system domain name.
client.keytab	Directory where the used keytab file is located when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	/opt/client/conf/loader.keytab
krb5.conf.file	Directory where the **krb5.conf** file is located when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	/opt/client/conf/krb5.conf

Table 20-132 Configuration parameters of job.properties

Configuration parameters	Description	Example Value
job.jobName	Job name.	job1
file.fileName.prefix	File name prefix.	table1
file.fileName.posfix	File name suffix.	.txt

Configuration parameters	Description	Example Value
file.filter	File filter, which filters files by matching file names. • **true** indicates that the preceding prefix or suffix is used to match all files in the input path. For details, see the example. • **false** indicates that the preceding prefix or suffix is used to match a file in the input path. For details, see the example.	true
date.day	Number of delayed days, which is matched with the date in the name of an imported file. For example, if the input date is 20160202 and the number of delayed days is 3, files that contain the 20160205 date field in the input path are matched. For details, see **schedule-tool Usage Example**.	3
file.date.format	Log format included in the name of the file to be imported.	yyyyMMdd
parameter.date.format	Entered date format when a script is invoked, which is usually consistent with **file.date.format**.	yyyyMMdd
file.format.iscompressed	Whether the file to be imported is a compressed file.	false
storage.type	Storage type. The final type of the file to be imported include HDFS, HBase, and Hive.	HDFS

NOTE

schedule-tool supports the configuration of multiple jobs at the same time. When multiple jobs are configured at the same time, **job.jobName**, **file.fileName.prefix**, and **file.fileName.posfix** in Table 20-132 need to be configured with multiple values, and the values need to be separated by **commas (,)**.

Precautions

server.url must be set to a format string of two IP addresses and port numbers, and the IP addresses and ports need to be separated by **commas (,)**.

20.9.5 schedule-tool Usage Example

Scenario

After a job is created using the Loader WebUI or Loader-tool, use schedule-tool to execute the job.

Prerequisites

The Loader client has been installed and configured. For details, see **Running a Loader Job by Using Commands**.

Procedure

Step 1 In the directory **/opt/houjt/test03** on the SFTP server, create multiple files with **table1** as the prefix, **.txt** as the suffix, and **yyyyMMdd** as the date format in the middle of the file name.

Figure 20-91 Example

```
[root@C12-RHEL64-ZYL111 test03]# ll
total 36
-rw-r--r--. 1 root root 54 Feb 29 19:11 table120160221.txt
-rw-r--r--. 1 root root 54 Feb 29 19:11 table120160222.txt
-rw-r--r--. 1 root root 54 Feb 29 19:11 table120160223.txt
-rw-r--r--. 1 root root 54 Feb 29 19:11 table120160224.txt
-rw-r--r--. 1 root root 54 Feb 29 19:11 table120160225.txt
-rw-r--r--. 1 root root 54 Feb 29 19:11 table120160226.txt
-rw-r--r--. 1 root root 54 Feb 29 18:43 table120160227.txt
-rw-r--r--. 1 root root 54 Feb 29 19:11 table120160228.txt
-rw-r--r--. 1 root root 54 Feb 29 19:11 table120160229.txt
```

Step 2 Create a Loader job of importing data from the SFTP server to HDFS. For details, see **Using Loader to Import Data from an SFTP Server to HDFS or OBS**.

Step 3 Log in to the node where the client is located as the user who installs the client.

Step 4 Run the following command to go to the **conf** directory of schedule-tool. For example, if the Loader client installation directory is **/opt/client/Loader/**, run the following command:

cd /opt/client/Loader/loader-tools-1.99.3/schedule-tool/conf

Step 5 Run the following command to edit the schedule.properties file and configure the login mode:

vi schedule.properties

schedule-tool supports two login modes. Only one mode can be selected. For parameter details, see **schedule-tool Usage Guide**. There can be security risks if a configuration file contains the authentication password. You are advised to delete the configuration file or use other secure methods to keep the password.

- When the password mode is used for login, the configuration information example is as follows:

```
[server.url = 10.10.26.187:21351,127.0.0.2:21351]
[authentication.type = kerberos]
```



```
[use.keytab = false]
[authentication.user = admin]
# Passwords stored in plaintext pose security risks. Store them in ciphertext in configuration files or environment variables.
[authentication.password= xxx]
```

- When the keytab file mode is used for login, the configuration information example is as follows:

```
[server.url = 10.10.26.187:21351,127.0.0.2:21351]
[authentication.type = kerberos]
[use.keytab = true]
[client.principal = bar]
[client.keytab = /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config/user.keytab]
[krb5.conf.file = /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config/krb5.conf]
```

Step 6 Run the following command to edit the job.properties file and configure job information:

vi job.properties

```
#job name
job.jobName = sftp2hdfs-schedule-tool


#Whether to update the loader configuration parameters(File filter)£?This parameter is used to match the import file name.Values are true or false.
#false means update.the file name which is get by schedule tool will be updated to Loader configuration parameters (File filter).
#false means no update.the file name which is get by schedule tool will be updated to Loader configuration parameters (import path).
file.filter = false


#File name = prefix + date + suffix
#Need to import the file name prefix
file.fileName.prefix=table1

#Need to import the file name suffixes
file.fileName.posfix=.txt

#Date Days.Value is an integer.
#According to the date and number of days to get the date of the import file.
date.day = 1

#Date Format.Import file name contains the date format.Format Type£ºyyyyMMdd,yyyyMMdd HHmmss,yyyy-MM-dd,yyyy-MM-dd HH:mm:ss
file.date.format = yyyyMMdd

#Date Format.Scheduling script execution. Enter the date format.
parameter.date.format = yyyyMMdd


#Whether the import file is a compressed format.Values ??are true or false.
#true indicates that the file is a compressed format£?Execution scheduling tool will extract the files.false indicates that the file is an uncompressed.Execution scheduling tool does not unpack.
file.format.iscompressed = false

#Hadoop storage type.Values are HDFS or HBase.
storage.type = HDFS
```

According to the data provided by **Step 1**, the filtering rules are set as follows when the **table120160221.txt** file is used as an example:

- File name prefix:

  file.fileName.prefix=table1
- File name suffix:

  file.fileName.posfix=.txt

- Date format included in the file name:

  file.date.format = yyyyMMdd
- Entered date parameter for invoking the script:

  parameter.date.format = yyyyMMdd
- Number of delayed days.

  date.day = 1

  For example, if the input date parameter of the script is **20160220**, the result is **20160221** by using the addition.

  NOTE

  If the **./run.sh 20160220 /user/loader/schedule_01** command is executed, the preceding filtering rules will be combined into a string: **"table1"+"20160221"+.txt = table120160221.txt**.

Step 7 Select a filtering rule according to the value of **file.filter**.

- If a file is to be exactly matched, go to **Step 8**.
- If a series of files are to be fuzzily matched, go to **Step 9**.

Step 8 Change the value of **file.filter** in the **job.properties** file to **false**.

Run the following commands to run the job. The task is completed.

cd /opt/client/Loader/loader-tools-1.99.3/schedule-tool

./run.sh *20160220 /user/loader/schedule_01*

20160220 indicates the input date, and */user/loader/schedule_01* indicates the output path.

NOTE

The string **table120160221.txt** obtained by combining the preceding filtering rules will be used as the file name and appended to the input path of the job. Therefore, the job will only process the uniquely matched file **table120160221.txt**.

Step 9 In the **job.properties** file, change the value of **file.filter** to **true**, and set the value of **file.fileName.prefix** to *****.

Run the following commands to run the job. The task is completed.

cd /opt/client/Loader/loader-tools-1.99.3/schedule-tool

./run.sh *20160220 /user/loader/schedule_01*

20160220 indicates the input date, and */user/loader/schedule_01* indicates the output path.

NOTE

The string ***20160221.txt** obtained by combining the preceding filtering rules will be used as the fuzzy match mode of the file filter. In the input path of the job, all files matching ***20160221.txt** will be processed by the job.

----End

20.9.6 Using loader-backup to Back Up Job Data

Scenario

After a job is created using the Loader WebUI or loader-tool, use loader-backup to back up data.

NOTE

- Only Loader jobs of data export support data backup.
- This tool is an internal Loader interface and is invoked by the upper-layer component HBase. Only the data backup from HDFS to SFTP is supported.

Prerequisites

The Loader client has been installed and configured. For details, see **Running a Loader Job by Using Commands**.

Procedure

Step 1 Log in to the node where the client is installed as the user who installs the client. For details, see **Running a Loader Job by Using Commands**.

Step 2 Run the following command to go to the directory where the **backup.properties** file is located. For example, if the Loader client installation directory is **/opt/client/Loader/**, run the following command:

cd /opt/client/Loader/loader-tools-1.99.3/loader-backup/conf

Step 3 Run the following command to modify the configuration parameters of **backup.properties**. For details about the parameters, see **Table 20-133**.

vi backup.properties

```
server.url = 10.0.0.1:21351,10.0.0.2:12000
authentication.type = kerberos
authentication.user =
authentication.password=
job.jobId = 1
use.keytab = true
client.principal = loader/hadoop
client.keytab = /opt/client/conf/loader.keytab
```


Table 20-133 Configuration parameters

Configuration parameters	Description	Example Value
server.url	Floating IP address and port (21351) for Loader. For compatibility, multiple IP addresses and ports can be configured and need to be separated by commas (,). The first IP address and port must be those of Loader (21351). The others can be configured based on service requirements.	10.0.0.1:21351,10.0.0.2:12000
authentication.type	Login authentication mode. • **kerberos** indicates that the security mode is used and Kerberos authentication is performed. Kerberos authentication provides two authentication modes: the password mode and the keytab file mode. • **simple** indicates that the normal mode is used and Kerberos authentication is not performed.	kerberos
authentication.user	User for login when the normal mode or password authentication is used. In the keytab login mode, this parameter does not need to be set.	bar

Configuration parameters	Description	Example Value
authentication.password	User password for login when the password authentication mode is used. In the normal mode or keytab login mode, this parameter does not need to be set. The password needs to be encrypted. The encryption method is described as follows: 1. Go to the directory where **encrypt_tool** is located. For example, if the Loader client installation directory is **/opt/hadoopclient/Loader**, run the following command: **cd /opt/hadoopclient/Loader/loader-tools-1.99.3** 2. Run the following command to encrypt the non-encrypted password. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage. **./encrypt_tool** *Unencrypted password* The obtained encrypted password is used as the value of **authentication.password**. **NOTE** If a non-encrypted password contains special characters, the special characters must be escaped. For example, the dollar sign ($) is a special character and can be escaped using single quotation marks ('). If a non-encrypted password contains single quotation marks, use double quotation marks to escape the single quotation marks. If a non-encrypted password contains double quotation marks, use backslashes (\) to escape the double quotation marks. For details, see the shell escape character rules.	-

Configuration parameters	Description	Example Value
job.jobId	ID of the job whose data is to be backed up. Job IDs can be viewed under created jobs on the Loader web UI.	1
use.keytab	Whether to use the keytab mode to log in. ● **true** indicates using the keytab file to log in. ● **false** indicates using the password to log in.	true
client.principal	User principal for accessing the Loader service when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	loader/hadoop
client.keytab	Directory where the used keytab file is located when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	/opt/client/conf/loader.keytab

Step 4 Run the following command to go to the directory where the backup script **run.sh** is located. For example, if the Loader client installation directory is **/opt/hadoopclient/Loader**, run the following command:

cd /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-backup

Step 5 Run the following command to run the backup script **run.sh** to back up Loader job data. The system backs up data to a directory at the same layer of the job output directory.

./run.sh *Backup data input directory*

For example, the backup data input directory is **/user/hbase/**, and the job output directory is**/opt/client/sftp/sftp1**. **sftp1** acts as a placeholder. Run the following command to back up data to the **/opt/client/sftp/hbase** directory:

./run.sh /user/hbase/

----End

20.9.7 Open Source sqoop-shell Tool Usage Guide

Overview

Sqoop-shell is a shell tool of Loader. All its functions are implemented by executing the **sqoop2-shell** script.

The sqoop-shell tool provides the following functions:

- Creating and updating connectors
- Creating and updating jobs
- Deleting connectors and jobs
- Starting jobs in the synchronous or asynchronous mode.
- Stopping jobs
- Viewing job status
- Viewing historical execution records of jobs
- Cloning connectors and jobs
- Creating and updating conversion steps
- Specifying line and field separators

The sqoop-shell tool supports the following modes:

- Interaction mode

  Users execute the **sqoop2-shell** script without parameters to go to the particular interaction window of Loader. After the contents of the script are input, the tool returns the relevant information to the interaction window.

- Batch mode

  The **sqoop2-shell** script has a file name as a parameter and multiple commands are stored in lines in the file. The sqoop-shell tool runs all commands in the file in sequence by executing the script. Alternatively, users can execute the **sqoop2-shell** script, to the end of which a command is attached with the **-c** parameter as the bridge. In this case, the sqoop-shell tool runs one command each time.

The sqoop-shell implements functions of Loader by running the commands in **Table 20-134**.

Table 20-134 Command list

Command	Description
exit	Exits the interaction mode. This command is supported only in the interaction mode.
history	Views the executed commands. This command is supported only in the interaction mode.
help	Views the tool help information.

Command	Description
set	Sets server attributes.
show	Displays service attributes and all the metadata information of Loader.
create	Creates connectors and jobs.
update	Updates connectors and jobs.
delete	Deletes connectors and jobs.
clone	Clones connectors and jobs.
start	Starts jobs.
stop	Stops jobs.
status	Views job status.

Commands

- The sqoop2-shell tool provides two methods to obtain login authentication information. The first method is to obtain login authentication information from the configuration file. For details about the configuration items, see **Importing Data to HDFS Using sqoop-shell** and **Importing Data to HDFS Using sqoop-shell**. The second one is to obtain the authentication information by using parameters. Two modes are available in the second method: password mode and Kerberos authentication mode.
- Command for accessing the interaction mode

  Execute the **sqoop2-shell** script without parameters to go to the sqoop tool window and run the commands one by one.

  Run the following command to obtain the authentication information by reading the configuration file:

  ./sqoop2-shell

  Run the following command to authenticate login using the password mode:

  ./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword*

  Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

  Run the following command to authenticate login using the Kerberos mode:

  ./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal*

  The following information is displayed:

  ```
  Welcome to sqoop client
  Use the username and password authentication mode
  Authentication success.
  ```



```
Sqoop Shell: Type 'help' or '\h' for help.

sqoop:000>
```

- Command for entering the batch mode

  Two methods are available for accessing the batch mode.

  - Execute the **sqoop2-shell** script, in which a file name is used as a parameter and multiple commands are stored in lines in this file. The sqoop-shell tool runs all commands in the file in sequence. The script must be stored in the home directory of the current user, for example, **/root/batchCommand.sh**.

    Run the following command to authenticate login by reading configuration files:

    ./sqoop2-shell */root/batchCommand.sh*

    Run the following command to authenticate login using the password mode:

    ./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword /root/batchCommand.sh*

    Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

    Run the following command to authenticate login using the Kerberos mode:

    ./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal /root/batchCommand.sh*

    batchCommand.sh is the user-defined name of the text file.

  - Execute the **sqoop2-shell** script, to the end of which a command is attached with the -c parameter as the bridge. The sqoop-shell tool will execute the command.

    Run the following command to authenticate login by reading configuration files:

    ./sqoop2-shell *-c expression*

    Run the following command to authenticate login using the password mode:

    ./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword -c expression*

    Run the following command to authenticate login using the Kerberos mode:

    ./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal -c expression*

    expression is the attached statement, whose format is the same as that in the text file in the first method.

- Exit command

  This command is used for exiting the interaction mode and supported only in the interaction mode.

  Example:

```
Welcome to sqoop client
Use the username and password authentication mode
Authentication success.
Sqoop Shell: Type 'help' or '\h' for help.
```



```
sqoop:000> exit
10-5-211-9:/opt/hadoopclient/Loader/loader-tools-1.99.3/sqoop-shell#
```

- History command

  This command is used for viewing the executed commands and supported only in the interaction mode.

  Example:

```
sqoop:000> history
  0  show connector
  1  create connection -c 4
  2  show connections;
  3  show connection;
  4  show connection -a;
  5  show connections;
  6  show connection;
  7  show connection -x 53;
  8  show connection -x 52;
  9  show connection -x 2
 10  show connection -x 53;
 11  show connection
 12  show connection -x 53
 13  create job -x 53 -t import
 14  show connector
 15  create connection -c 5
 16  show connection -x 54
 17  exit
 18  show connector
 19  create connection -c 5
 20  exit
 21  show connector
 22  create connection -c 6
 23  create job -x 20 -t import
 24  start job -j 85 -s
 25  \x
 26  exit
 27  history
sqoop:000>
```

- Help command

  This command is used for viewing the tool help information.

  Example:

```
sqoop:000> help
For information about Sqoop, visit: http://sqoop.apache.org/docs/1.99.3/index.html

Available commands:
  exit    (\x  ) Exit the shell
  history (\H  ) Display, manage and recall edit-line history
  help    (\h  ) Display this help message
  set     (\st ) Set server or option Info
  show    (\sh ) Show server, connector, framework, connection, job, submission or option Info
  create  (\cr ) Create connection or job Info
  delete  (\d  ) Delete connection or job Info
  update  (\up ) Update connection or job Info
  clone   (\cl ) Clone connection or job Info
  start   (\sta) Start job
  stop    (\stp) Stop job
  status  (\stu) Status job

For help on a specific command type: help command

sqoop:000>
```

- Set command

  The set command is used for setting attributes of clients and servers and supports the following attributes:

– **server** indicates setting the connection attributes for servers.

  NOTE

  When attribute -u is set, attributes -h, -p, and -w can be ignored.

– **option** indicates setting the client attributes.

  NOTE

  option can be set by key values. For example, **set option --name verbose --value true**.

Attribute Type	Subattribute	Description
server	-h,--host	Service IP address.
	-p,--port	Service Port
	-w,--webapp	Tomcat application name.
	-u,--url	Sqoop service URL.
option	verbose	Redundancy mode, which indicates that more information is printed.
	poll-timeout	Sets the polling timeout duration.

Example:

```
set option --name verbose --value false
set server --host 10.0.0.1 --port 21351 --webapp loader
```

- **show** command

  This command is used for displaying information, such as variable information and storage metadata information.

Attribute Type	Subattribute	Description
server	-a,--all	Displays all server attributes.
	-p,--port	Displays the service port.
	-w,--webapp	Displays the Tomcat application name.
	-h,--host	Displays the service IP address.
option	-name	Displays the attributes of the specified name.
connector	-a,--all	Displays information about all connection types.

Attribute Type	Subattribute	Description
	-c,--cid	Displays information about the connection type of a specified ID.
framework	None.	Displays metadata information about frameworks.
connection	-a,--all	Displays all connection attributes.
	-x,--xid	Displays the attributes of a specified connection.
	-n,--name	Displays the connection attributes of a specified name.
job	-a,--all	Displays information about all jobs.
	-j,--jid	Displays job information about a specified ID.
	-n,--name	Displays job information about a specified name.
submission	-j,--jid	Displays the submission record of a specified job.
	-d,--detail	Displays details.

Example:

```
show server -all
show option --name verbose
show connector -all
show framework
show connection -all
show connection -n sftp-example
show job -all
show job -j 1
show submission --jid 1
show submission --jid 1 -d
```

- Create command

  This command is used for creating connectors and jobs.

Attribute Type	Subattribute	Description
connection	-c,--cid	Specifies the ID of a connector type.

Attribute Type	Subattribute	Description
	-cn,--cname	Specifies the name of a specified connector type.
job	-x,--xid	Specifies the connector ID.
	-xn,--xname	Specifies the connector name.
	-t,--type	Specifies the job type. Possible values: ● import ● export

- In the interaction mode, enter the attribute values one by one as prompted.

  Example for creating connectors:

  ```
  create connection -c 1
  create connection -cn example
  ```

  Example for creating jobs:

  ```
  create job -x 1 -t import
  create job -xn job_example -t export
  ```

- In the batch mode, run the following command to view the specific attribute and then set a value for the attribute:

  create job -t import -x 1 --help

  You can run the above command in either of the following ways:

  Save the command to a text file and attach this file to the end of the **sqoop-shell** script, and run the following command:

  ./sqoop2-shell *batchCommand.sh*

  Attach a command with the **-c** parameter to the end of the **sqoop-shell** script and run the following command:

  ./sqoop2-shell *-c expression*

  For details about command execution, refer to previous description in this section. The following shows two complete commands:

  Example for creating connectors:

  ```
  create connection -c 4 --connector-connection-sftpPassword xxxxx --connector-connection-sftpServerIp 10.0.0.1 --connector-connection-sftpServerPort 22 --connector-connection-sftpUser root--name testConnection
  ```

  Example for creating jobs:

  ```
  create job -t import -x 1 --connector-file-inputPath /opt/tempfile --connector-file-fileFilter * --framework-output-outputDirectory /user/loader/1 --framework-output-storageType HDFS --framework-throttling-extractorSize 120 --framework-output-fileType TEXT_FILE --connector-file-splitType FILE -queue default -priority low -name newJob
  ```

- In the batch mode, you can attach a statement using the **-c** parameter as the bridge.

  Example for creating connectors:


```
./sqoop2-shell -c "create connection -c 4 --connector-connection-sftpPassword xxxxx --connector-connection-sftpServerIp 10.0.0.1 --connector-connection-sftpServerPort 22 --connector-connection-sftpUser root--name testConnection"
```

- **update** command

  This command is used for updating connectors and jobs.

Attribute Type	Subattribute	Description
connection	-x,--xid	Specifies the connector ID. **NOTE** When the connectors are updated, the password must be set.
job	-j,--jid	Specifies the job ID.

  - Interaction mode

    Example for updating connectors:

```
update connection --xid 1
```

    Example for updating jobs:

```
update job --jid 1
```

  - Batch mode

    Example for updating connectors:

```
update connection -x 6 --connector-connection-sftpServerPort 21 - --name sfp_130--connector-connection-sftpPassword xxxx
```

    Example for updating jobs:

    Example 1:

```
update job -jid 1 -name sftp2hdfs --connector-file-fileFilter *.txt
```

    Example 2:

```
./sqoop2-shell -uk true -k /opt/loader/user.keytab -s luser /opt/loader/testupdate.txt
./sqoop2-shell -uk true -k /opt/loader/user.keytab -s luser -c "update job --jid 24 --name oracle-hive --connector-table-sql 'SELECT * FROM range_example WHERE replace(datadt,\'-\',\'\')='20240801' and \${CONDITIONS}'"
```

    NOTE

    When updating a job, you can write the update commands in a file, for example, **/opt/loader/testupdate.txt** (the file name can be customized), or specify the commands using **--connector-table-sql**, in which the **sql** command must be enclosed in single quotation marks ('). For details, see example 2. Involved commands include **connector-table-sql**, **connector-table-columns**, **connector-table-partitionColumn**, **connector-table-conditions**, **connector-table-queryCondition**.

- **delete** command

  This command is used for deleting connectors and jobs.

Attribute Type	Subattribute	Description
connection	-x,--xid	Specifies the connector ID.

Attribute Type	Subattribute	Description
	-n,--name	Specifies the connector name.
job	-j,--jid	Specifies the job ID.
	-n,--name	Specifies the job name.

Example:

```
delete connection -x 1
delete connection --name abc
delete job -j 1
delete job -n qwerty
```

- **clone** command

  This command is used for cloning connectors and jobs.

Attribute Type	Subattribute	Description
connection	-x,--xid	Specifies the connector ID. **NOTE** The password and connector name must be entered when the connectors are cloned.
job	-j,--jid	Specifies the job ID.

Example:

```
clone job -j 1
```

- **start** command

  This command is used for starting jobs.

Attribute Type	Subattribute	Description
job	-j,--jid	Specifies the job ID.
	-n,--name	Specifies the job name.
	-s,--synchronous	Whether to start jobs in the synchronous mode or not.

Example for starting jobs in the asynchronous mode:

```
start job -j 1
start job -n abc
```

Example for starting jobs in the synchronous mode:


```
start job -j 1 -s
start job --name abc --synchronous
```

- **stop** command

  This command is used for stopping jobs.

Attribute Type	Subattribute	Description
job	-j,--jid	Specifies the job ID.
	-n,--name	Specifies the job name.

  Example:

```
stop job -j 1
stop job -n abc
```

- Status command

  This command is used for viewing job status.

Attribute Type	Subattribute	Description
job	-j,--jid	Specifies the job ID.

  When **-s** parameter is attached to the command, the result only contains the enumerated value of job status.

  Example:

```
status job -j 1
status job -j 1 -s
```

Extended Attributes of Create Command

For the scenario in which HDFS exchanges data with the SFTP server or RDB, MRS extends the create command attributes on the basis of the open source sqoop-shell tool, so as to specify line and field separators and conversion steps when jobs are created.

Table 20-135 Extended Attributes of Create Command

Property	Description
fields-terminated-by	Default field separator.
lines-terminated-by	Default line separator.
input-fields-terminated-by	Inputs the step field separator. If the step field separator is not specified, the value equals to **fields-terminated-by** by default.

Property	Description
input-lines-terminated-by	Inputs the step line separator. If the step line separator is not specified, the value equals to **lines-terminated-by** by default.
output-fields-terminated-by	Outputs the step field separator. If the step field separator is not specified, the value equals to **fields-terminated-by** by default.
output-lines-terminated-by	Outputs the step line separator. If the step line separator is not specified, the value equals to **lines-terminated-by** by default.
trans	Specifies the conversion steps. The value is the directory where the conversion step file is located. When the relative directory of file is specified, the file is by default stored in the directory where the **sqoop2-shell** script is located. When the attribute is set, the other extended attributes can be ignored.

Interconnecting Sqoop1 with MRS

Step 1 Download the open source Sqoop from **http://www.apache.org/dyn/closer.lua/sqoo:p/1.4.7**.

Step 2 Save the downloaded **sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz** package to the **/opt/sqoop** directory on the Master node in the MRS cluster and decompress the package.

tar zxvf sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz

Step 3 Go to the directory where the package is decompressed and modify the configuration.

cd /opt/sqoop/sqoop-1.4.7.bin__hadoop-2.6.0/conf

cp sqoop-env-template.sh sqoop-env.sh

vi sqoop-env.sh

Add the following configurations:

```
export HADOOP_COMMON_HOME=/opt/client/HDFS/hadoop
export HADOOP_MAPRED_HOME=/opt/client/HDFS/hadoop
export HIVE_HOME=/opt/Bigdata/MRS_1.9.X/install/FusionInsight-Hive-3.1.0/hive (Enter the actual path.)
export HIVE_CONF_DIR=/opt/client/Hive/config
export HCAT_HOME=/opt/client/Hive/HCatalog
```

Step 4 Add the system variable **SQOOP_HOME** to **PATH**.

vi /etc/profile

Add the following information:

```
export SQOOP_HOME=/opt/sqoop/sqoop-1.4.7.bin__hadoop-2.6.0
export PATH=$PATH:$SQOOP_HOME/bin
```

Step 5 Run the following command to copy the **jline-2.12.jar** file to the **lib** file.

cp /opt/share/jline-2.12/jline-2.12.jar /opt/sqoop/sqoop-1.4.7.bin__hadoop-2.6.0/lib

Step 6 Run the following command to add the following configuration to the file.

vim $JAVA_HOME/jre/lib/security/java.policy

```
permission javax.management.MBeanTrustPermission "register";
```

Step 7 Run the following command to interconnect sqoop1 with MRS.

source /etc/profile

----End

20.9.8 Importing Data to HDFS Using sqoop-shell

Scenario

Taking importing data from SFTP to HDFS as an example, this section introduces how to use the sqoop-shell tool to create and start Loader jobs in the interaction mode and batch mode.

Prerequisites

The Loader client has been installed and configured. For details, see **Running a Loader Job by Using Commands**.

Example for the Interaction Mode

Step 1 Log in to the node where the Loader client is installed as the user who installs the client.

Step 2 Run the following command to go to the **conf** directory of the sqoop-shell tool. For example, if the Loader client installation directory is **/opt/client/Loader/**, run the following command:

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell/conf

Step 3 Run the following command to configure authentication information:

vi client.properties

```
server.url=10.0.0.1:21351
# simple or kerberos
authentication.type=simple
# true or false
use.keytab=true

authentication.user=
authentication.password=

client.principal=hdfs/hadoop@<system domain name>
```



```
# keytab file
client.keytab.file=./conf/login/hdfs.keytab
```

NOTE

Log in to FusionInsight Manager and choose **System** > **Permission** > **Domain and Mutual Trust**. The value of **Local Domain** is the current system domain name.

Table 20-136 Configuration parameters

Configuration parameters	Description	Example Value
server.url	Floating IP address and port (21351) for Loader. For compatibility, multiple IP addresses and ports can be configured and need to be separated by commas (**,**). The first IP address and port must be those of Loader (21351). The others can be configured based on service requirements.	10.0.0.1:21351
authentication.type	Login authentication mode. • **kerberos** indicates that the security mode is used and Kerberos authentication is performed. Kerberos authentication provides two authentication modes: the password mode and the keytab file mode. • **simple** indicates that the normal mode is used and Kerberos authentication is not performed.	kerberos
authentication.user	User for login when the normal mode or password authentication is used. In the keytab login mode, this parameter does not need to be set.	bar

Configuration parameters	Description	Example Value
authentication.password	User password for login when the password authentication mode is used. In the normal mode or keytab login mode, this parameter does not need to be set. The password needs to be encrypted. The encryption method is described as follows: 1. Go to the directory where **encrypt_tool** is located. For example, if the Loader client installation directory is **/opt/hadoopclient/Loader**, run the following command: **cd /opt/hadoopclient/Loader/loader-tools-1.99.3** 2. Run the following command to encrypt the non-encrypted password. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage. **./encrypt_tool** *Unencrypted password* The obtained encrypted password is used as the value of **authentication.password**. **NOTE** If a non-encrypted password contains special characters, the special characters must be escaped. For example, the dollar sign ($) is a special character and can be escaped using single quotation marks ('). If a non-encrypted password contains single quotation marks, use double quotation marks to escape the single quotation marks. If a non-encrypted password contains double quotation marks, use backslashes (\) to escape the double quotation marks. For details, see the shell escape character rules.	-

Configuration parameters	Description	Example Value
use.keytab	Whether to use the keytab mode to log in. • **true** indicates using the keytab file to log in. • **false** indicates using the password to log in.	true
client.principal	User principal for accessing the Loader service when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	loader/hadoop
client.keytab.file	Directory where the used keytab file is located when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	/opt/client/conf/loader.keytab

Step 4 Run the following command to go to the interaction mode:

source /opt/client/bigdata_env

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell

./sqoop2-shell

The preceding commands obtain authentication information by reading the configuration file.

Alternatively, you can also use the password or Kerberos authentication.

Run the following command to authenticate login using the password mode:

./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword*

Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

Run the following command to authenticate login using the Kerberos mode:

./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal*

```
Welcome to sqoop client
Use the username and password authentication mode
Authentication success.
Sqoop Shell: Type 'help' or '\h' for help.

sqoop:000>
```

Step 5 Run the following command to view the corresponding ID of the current connector:

show connector

The following information is displayed:

```
+----+----------------------------+----------------+-----------------------------------------------------------------------+
| Id |            Name            |    Version     |                                Class                                  |
+----+----------------------------+----------------+-----------------------------------------------------------------------+
| 1  | generic-jdbc-connector     | 2.0.6-SNAPSHOT |
org.apache.sqoop.connector.jdbc.GenericJdbcConnector                |
| 2  | ftp-connector              | 2.0.5-SNAPSHOT |
org.apache.sqoop.connector.ftp.FtpConnector                         |
| 3  | hdfs-connector             | 2.0.5-SNAPSHOT |
org.apache.sqoop.connector.hdfs.HdfsConnector                       |
| 4  | oracle-connector           | 2.0.1-SNAPSHOT |
org.apache.sqoop.connector.oracle.OracleConnector                   |
| 5  | mysql-fastpath-connector   | 2.0.1-SNAPSHOT |
org.apache.sqoop.connector.mysql.MySqlConnector                     |
| 6  | sftp-connector             | 2.0.5-SNAPSHOT |
org.apache.sqoop.connector.sftp.SftpConnector                       |
| 7  | oracle-partition-connector | 2.0.6-SNAPSHOT |
org.apache.sqoop.connector.oracle.partition.OraclePartitionConnector |
+----+----------------------------+----------------+-----------------------------------------------------------------------+
```

The preceding information indicates that the SFTP connector ID is 6.

Step 6 Run the following command to create connectors and enter the specific connector information as prompted:

create connection -c *connector ID*

For example, if the connector ID is 6, run the following command:

create connection -c 6

```
sqoop:000> create connection -c 6
Creating connection for connector with id 6
Please fill following values to create new connection object
Name: sftp14

Connection configuration

Sftp server IP: 10.0.0.1
Sftp server port: 22
Sftp user name: root
Sftp password: ************
Sftp public key:
New connection was successfully created with validation status FINE and persistent id 20
sqoop:000>
```

The preceding information indicates that the connection ID is 20.

Step 7 Based on the connection ID, run the following command to create jobs:

create job -x *connection ID* **-t import**

For example, if the connection ID is 20, run the following command:

create job -x 20 -t import

The following information is displayed:

```
Creating job for connection with id 20
Please fill following values to create new job object
Name: sftp-hdfs-test

File configuration

Input path: /opt/tempfile
```



```
File split type:
  0 : FILE
  1 : SIZE
Choose: 0
Filter type:
  0 : WILDCARD
  1 : REGEX
Choose: 0
Path filter: *
File filter: *
Encode type:
Suffix name:
Compression:

Output configuration

Storage type:
  0 : HDFS
  1 : HBASE_BULKLOAD
  2 : HBASE_PUTLIST
  3 : HIVE
Choose: 0
File type:
  0 : TEXT_FILE
  1 : SEQUENCE_FILE
  2 : BINARY_FILE
Choose: 0
Compression format:
  0 : NONE
  1 : DEFAULT
  2 : DEFLATE
  3 : GZIP
  4 : BZIP2
  5 : LZ4
  6 : SNAPPY
Choose:
Output directory: /user/loader/test
File operate type:
  0 : OVERRIDE
  1 : RENAME
  2 : APPEND
  3 : IGNORE
  4 : ERROR
Choose: 0

Throttling resources

Extractors: 2
Extractor size:
New job was successfully created with validation status FINE  and persistent id 85
sqoop:000>
```

The preceding information indicates that the job ID is 85.

Step 8 Run the following command to start the job:

start job -j *job ID* **-s**

For example, if the job ID is 85, run the following command:

start job -j 85 -s

Displaying the **SUCCEEDED** information indicates that the job is started successfully.

```
Submission details
Job ID: 85
Server URL: https://10.0.0.0:21351/loader/
Created by: admin
Creation date: 2016-07-20 16:25:38 GMT+08:00
```



```
Lastly updated by: admin
2016-07-20 16:25:38 GMT+08:00: BOOTING  - Progress is not available
2016-07-20 16:25:46 GMT+08:00: BOOTING  - 0.00 %
2016-07-20 16:25:53 GMT+08:00: BOOTING  - 0.00 %
2016-07-20 16:26:08 GMT+08:00: RUNNING  - 90.00 %
2016-07-20 16:26:08 GMT+08:00: RUNNING  - 90.00 %
2016-07-20 16:26:27 GMT+08:00: SUCCEEDED
```

----End

Example for the Batch Mode

Step 1 Log in to the node where the Loader client is installed as the user who installs the client.

Step 2 Run the following command to go to the **conf** directory of the sqoop-shell tool. For example, if the Loader client installation directory is **/opt/client/Loader/**, run the following command:

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell/conf

Step 3 Run the following command to configure authentication information:

vi client.properties

```
server.url=10.0.0.1:21351
# simple or kerberos
authentication.type=simple
# true or false
use.keytab=true

authentication.user=
authentication.password=

client.principal=hdfs/hadoop@<system domain name>

# keytab file
client.keytab.file=./conf/login/hdfs.keytab
```

Table 20-137 Configuration parameters

Configuration parameters	Description	Example Value
server.url	Floating IP address and port (21351) for Loader. For compatibility, multiple IP addresses and ports can be configured and need to be separated by commas (**,**). The first IP address and port must be those of Loader (21351). The others can be configured based on service requirements.	10.0.0.1:21351

Configuration parameters	Description	Example Value
authentication.type	Login authentication mode. • **kerberos** indicates that the security mode is used and Kerberos authentication is performed. Kerberos authentication provides two authentication modes: the password mode and the keytab file mode. • **simple** indicates that the normal mode is used and Kerberos authentication is not performed.	kerberos
authentication.user	User for login when the normal mode or password authentication is used. In the keytab login mode, this parameter does not need to be set.	bar

Configuration parameters	Description	Example Value
authentication.password	User password for login when the password authentication mode is used. In the normal mode or keytab login mode, this parameter does not need to be set. The password needs to be encrypted. The encryption method is described as follows: 1. Go to the directory where **encrypt_tool** is located. For example, if the Loader client installation directory is **/opt/hadoopclient/Loader**, run the following command: **cd /opt/hadoopclient/Loader/loader-tools-1.99.3** 2. Run the following command to encrypt the non-encrypted password: **./encrypt_tool** *Unencrypted password* The obtained encrypted password is used as the value of **authentication.password**. **NOTE** If a non-encrypted password contains special characters, the special characters must be escaped. For example, the dollar sign ($) is a special character and can be escaped using single quotation marks ('). If a non-encrypted password contains single quotation marks, use double quotation marks to escape the single quotation marks. If a non-encrypted password contains double quotation marks, use backslashes (\) to escape the double quotation marks. For details, see the shell escape character rules.	-
use.keytab	Whether to use the keytab mode to log in. ● **true** indicates using the keytab file to log in. ● **false** indicates using the password to log in.	true

Configuration parameters	Description	Example Value
client.principal	User principal for accessing the Loader service when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	loader/hadoop
client.keytab.file	Directory where the used keytab file is located when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	/opt/client/conf/loader.keytab

Step 4 Run the following command to go to the directory where the **sqoop2-shell** script is located and create a text file in the directory, such as **batchCommand.sh**:

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell

vi batchCommand.sh

An example of **batchCommand.sh** is displayed as follows:

```
View parameters
create connection -c 6 --help

// Create a connector
create connection -c 6 -name sftp-connection --connector-connection-sftpServerIp 10.0.0.1 --connector-connection-sftpServerPort 22 --connector-connection-sftpUser root --connector-connection-sftpPassword xxxxx

Create a job
create job -t import -x 20 --connector-file-inputPath /opt/tempfile --connector-file-fileFilter * --framework-output-outputDirectory /user/loader/1 --framework-output-storageType HDFS --framework-throttling-extractorSize 120 --framework-output-fileType TEXT_FILE --connector-file-splitType FILE -name test

Start a job
start job -j 85 -s
```

xxxxx is the password for the connector.

Step 5 Run the following command and the sqoop-shell tool will run the preceding commands in sequence:

./sqoop2-shell batchCommand.sh

The commands above authenticate login by reading configuration files. Alternatively, you can attach the authentication information to the command, that is, use the password mode or Kerberos mode to authenticate login.

Run the following command to authenticate login using the password mode:

./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword* **batchCommand.sh**

Run the following command to authenticate login using the Kerberos mode:

./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal* **batchCommand.sh**

Displaying the **SUCCEEDED** information indicates that the job is started successfully.

```
Welcome to sqoop client
Use the username and password authentication mode
Authentication success.
sqoop:000> create connection -c 6 --help
usage: Show connection parameters:
   --connector-connection-sftpPassword <arg>
   --connector-connection-sftpServerIp <arg>
   --connector-connection-sftpServerPort <arg>
   --connector-connection-sftpUser <arg>
   --framework-security-maxConnections <arg>
   --name <arg>
===> FINE
sqoop:000> create connection -c 6 -name sftp-connection --connector-connection-sftpServerIp 10.0.0.1 --
connector-connection-sftpServerPort 22 --connector-connection-sftpUser root --connector-connection-
sftpPassword xxxxx
Creating connection for connector with id 6
New connection was successfully created with validation status FINE and persistent id 20
===> FINE
sqoop:000> create job -t import -x 20 --connector-file-inputPath /opt/tempfile --connector-file-fileFilter * --
framework-output-outputDirectory /user/loader/1 --framework-output-storageType HDFS --framework-
throttling-extractorSize 120 --framework-output-fileType TEXT_FILE --connector-file-splitType FILE -name
test
Creating job for connection with id 20
New job was successfully created with validation status FINE  and persistent id 85
===> FINE

Submission details
Job ID: 85
Server URL: https://10.0.0.0:21351/loader/
Created by: admin
Creation date: 2016-07-20 16:25:38 GMT+08:00
Lastly updated by: admin
2016-07-20 16:25:38 GMT+08:00: BOOTING  - Progress is not available
2016-07-20 16:25:46 GMT+08:00: BOOTING  - 0.00 %
2016-07-20 16:25:53 GMT+08:00: BOOTING  - 0.00 %
2016-07-20 16:26:08 GMT+08:00: RUNNING  - 90.00 %
2016-07-20 16:26:08 GMT+08:00: RUNNING  - 90.00 %
2016-07-20 16:26:27 GMT+08:00: SUCCEEDED
```

Step 6 In the batch mode, the **-c** parameter can be used to attach a command. sqoop-shell can execute only the attached command at a time.

Run the following command to create a connection:

./sqoop2-shell -c "create connection -c 6 -name sftp-connection --connector-connection-sftpServerIp 10.0.0.1 --connector-connection-sftpServerPort 22 --connector-connection-sftpUser root --connector-connection-sftpPassword xxxxx"

You can also use the password mode or Kerberos mode to attach the authentication information to the command.

Run the following command to authenticate login using the password mode:

./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword* **-c "create connection -c 6 -name sftp-connection --connector-connection-sftpServerIp 10.0.0.1 --connector-connection-sftpServerPort 22 --connector-connection-sftpUser root --connector-connection-sftpPassword xxxxx"**

Run the following command to authenticate login using the Kerberos mode:

./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal* **-c "create connection -c 6 -name sftp-connection --connector-connection-sftpServerIp 10.0.0.1 --**

connector-connection-sftpServerPort 22 --connector-connection-sftpUser root --connector-connection-sftpPassword xxxxx"

Displaying the **FINE** information indicates the connection is created successfully.

```
Welcome to sqoop client
Use the username and password authentication mode
Authentication success.
sqoop:000> create connection -c 6 -name sftp-connection --connector-connection-sftpServerIp 10.0.0.1 --connector-connection-sftpServerPort 22 --connector-connection-sftpUser root --connector-connection-sftpPassword xxxxx
Creating connection for connector with id 6
New connection was successfully created with validation status FINE and persistent id 20
===> FINE
```

----End

20.9.9 Importing Data to HDFS Using sqoop-shell

Scenario

Taking **Importing Data from Oracle to HBase** as an example, this section introduces how to use the sqoop-shell tool to create and start Loader jobs in the interaction mode and batch mode.

Prerequisites

The Loader client has been installed and configured. For details, see **Running a Loader Job by Using Commands**.

Example for the Interaction Mode

Step 1 Log in to the node where the Loader client is installed as the user who installs the client.

Step 2 Run the following command to go to the **conf** directory of the sqoop-shell tool. For example, if the Loader client installation directory is **/opt/client/Loader/**, run the following command:

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell/conf

Step 3 Run the following command to configure authentication information:

vi client.properties

```
server.url=10.0.0.1:21351
# simple or kerberos
authentication.type=simple
# true or false
use.keytab=true

authentication.user=
authentication.password=

client.principal=oracle/hadoop@<system domain name>

# keytab file
client.keytab.file=./conf/login/oracle.keytab
```


NOTE

Log in to FusionInsight Manager and choose **System** > **Permission** > **Domain and Mutual Trust**. The value of **Local Domain** is the current system domain name.

Table 20-138 Configuration parameters

Configuration parameters	Description	Example Value
server.url	Floating IP address and port (21351) for Loader. For compatibility, multiple IP addresses and ports can be configured and need to be separated by **commas (,)**. The first IP address and port must be those of Loader (21351). The others can be configured based on service requirements.	10.0.0.1:21351
authentication.type	Login authentication mode. • **kerberos** indicates that the security mode is used and Kerberos authentication is performed. Kerberos authentication provides two authentication modes: the password mode and the keytab file mode. • **simple** indicates that the normal mode is used and Kerberos authentication is not performed.	kerberos
authentication.user	User for login when the normal mode or password authentication is used. In the keytab login mode, this parameter does not need to be set.	bar

Configuration parameters	Description	Example Value
authentication.password	User password for login when the password authentication mode is used. In the normal mode or keytab login mode, this parameter does not need to be set. The password needs to be encrypted. The encryption method is described as follows: 1. Go to the directory where **encrypt_tool** is located. For example, if the Loader client installation directory is **/opt/hadoopclient/Loader**, run the following command: **cd /opt/hadoopclient/Loader/loader-tools-1.99.3** 2. Run the following command to encrypt the non-encrypted password. Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage. **./encrypt_tool** *Unencrypted password* The obtained encrypted password is used as the value of **authentication.password**. **NOTE** If a non-encrypted password contains special characters, the special characters must be escaped. For example, the dollar sign ($) is a special character and can be escaped using single quotation marks ('). If a non-encrypted password contains single quotation marks, use double quotation marks to escape the single quotation marks. If a non-encrypted password contains double quotation marks, use backslashes (\) to escape the double quotation marks. For details, see the shell escape character rules.	-

Configuration parameters	Description	Example Value
use.keytab	Whether to use the keytab mode to log in. • **true** indicates using the keytab file to log in. • **false** indicates using the password to log in.	true
client.principal	User principal for accessing the Loader service when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	loader/hadoop
client.keytab.file	Directory where the used keytab file is located when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	/opt/client/conf/loader.keytab

Step 4 Run the following command to go to the interaction mode:

source /opt/client/bigdata_env

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell

./sqoop2-shell

The preceding commands obtain authentication information by reading the configuration file.

Alternatively, you can also use the password or Kerberos authentication.

Run the following command to authenticate login using the password mode:

./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword*

Commands containing authentication passwords pose security risks. Disable the command recording function (history) before running such commands to prevent information leakage.

Run the following command to authenticate login using the Kerberos mode:

./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal*

```
Welcome to sqoop client
Use the username and password authentication mode
Authentication success.
Sqoop Shell: Type 'help' or '\h' for help.

sqoop:000>
```

Step 5 Run the following command to view the corresponding ID of the current connector:

show connector

The following information is displayed:

```
| Id |            Name            |    Version     |                                 Class                                 |
+----+----------------------------+----------------+-----------------------------------------------------------------------+
| 1  | generic-jdbc-connector     | 2.0.7-SNAPSHOT |
org.apache.sqoop.connector.jdbc.GenericJdbcConnector                |
| 2  | ftp-connector              | 2.0.5-SNAPSHOT |
org.apache.sqoop.connector.ftp.FtpConnector                         |
| 3  | hdfs-connector             | 2.0.5-SNAPSHOT |
org.apache.sqoop.connector.hdfs.HdfsConnector                       |
| 4  | oracle-connector           | 2.0.1-SNAPSHOT |
org.apache.sqoop.connector.oracle.OracleConnector                   |
| 5  | mysql-fastpath-connector   | 2.0.1-SNAPSHOT |
org.apache.sqoop.connector.mysql.MySqlConnector                     |
| 6  | sftp-connector             | 2.0.6-SNAPSHOT |
org.apache.sqoop.connector.sftp.SftpConnector                       |
| 7  | oracle-partition-connector | 2.0.6-SNAPSHOT |
org.apache.sqoop.connector.oracle.partition.OraclePartitionConnector |
+----+----------------------------+----------------+-----------------------------------------------------------------------+
```

The preceding information indicates that the Oracle connector ID is 4.

Step 6 Run the following command to create connectors and enter the specific connector information as prompted:

create connection -c *connector ID*

For example, if the connector ID is 4, run the following command:

create connection -c 4

```
sqoop:000> create connection -c 4
Creating connection for connector with id 4
Please fill following values to create new connection object
Name: oracle14

Oracle connection configuration

JDBC connection string: jdbc:oracle:thin:@189.120.84.106:1521:orcl
Username: oracledba
Password: **********
JDBC connection properties:
There are currently 0 values in the map:
entry#
New connection was successfully created with validation status FINE and persistent id 3
sqoop:000>
```

The preceding information indicates that the connection ID is 3.

Step 7 Based on the connection ID, run the following command to create jobs:

create job -x *connection ID* **-t import --trans** *absolute path of job-config*/**oracle-hbase.json**

For example, if the connection ID is 3, run the following command:

create job -x 3 -t import --trans /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/oracle-hbase.json

The following information is displayed:

```
sqoop:000> create job -x 3 -t import --trans /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/oracle-to-hbase.json
Creating job for connection with id 3
Please fill following values to create new job object
```



```
Name: run

Database target

Table name: test
Columns:
Conditions:
Data split method:
  0 : ROWID
  1 : PARTITION
Choose:
Table Partitions:
Data split allocation method:
  0 : ROUNDROBIN
  1 : SEQUENTIAL
  2 : RANDOM
Choose:
JDBC fetch size:

Output configuration

Storage type:
  0 : HDFS
  1 : HBASE_BULKLOAD
  2 : HBASE_PUTLIST
  3 : HIVE
  4 : SPARK
Choose: 1
HBase instance: HBase
Clear data before import : false

Throttling resources

Extractors: 10
Extractor size:
New job was successfully created with validation status FINE  and persistent id 7
sqoop:000>
```

The preceding information indicates that the job ID is 7.

Step 8 Run the following command to start the job:

start job -j *job ID* **-s**

For example, if the job ID is 7, run the following command:

start job -j 7 -s

Displaying the **SUCCEEDED** information indicates that the job is started successfully.

```
Submission details
Job ID: 7
Server URL: https://10.0.0.0:21351/loader/
Created by: admintest
Creation date: 2019-12-04 16:37:34 CST
Lastly updated by: admintest
2019-12-04 16:37:34 CST: BOOTING  - Progress is not available
2019-12-04 16:37:42 CST: BOOTING  - 0.00 %
2019-12-04 16:37:42 CST: BOOTING  - 0.00 %
2019-12-04 16:37:57 CST: RUNNING  - 0.00 %
2019-12-04 16:38:12 CST: RUNNING  - 45.00 %
2019-12-04 16:38:12 CST: RUNNING  - 45.00 %
2019-12-04 16:38:27 CST: SUCCEEDED
```

----End

Example for the Batch Mode

Step 1 Log in to the node where the Loader client is installed as the user who installs the client.

Step 2 Run the following command to go to the **conf** directory of the sqoop-shell tool. For example, if the Loader client installation directory is **/opt/client/Loader/**, run the following command:

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell/conf

Step 3 Run the following command to configure authentication information:

vi client.properties

```
server.url=10.0.0.1:21351
# simple or kerberos
authentication.type=simple
# true or false
use.keytab=true

authentication.user=
authentication.password=

client.principal=hdfs/hadoop.@<system domain name>@<system domain name>

# keytab file
client.keytab.file=./conf/login/hdfs.keytab
```

Table 20-139 Configuration parameters

Configuration parameters	Description	Example Value
server.url	Floating IP address and port (21351) for Loader. For compatibility, multiple IP addresses and ports can be configured and need to be separated by **commas (,)**. The first IP address and port must be those of Loader (21351). The others can be configured based on service requirements.	10.0.0.1:21351
authentication.type	Login authentication mode. • **kerberos** indicates that the security mode is used and Kerberos authentication is performed. Kerberos authentication provides two authentication modes: the password mode and the keytab file mode. • **simple** indicates that the normal mode is used and Kerberos authentication is not performed.	kerberos

Configuration parameters	Description	Example Value
authentication.user	User for login when the normal mode or password authentication is used. In the keytab login mode, this parameter does not need to be set.	bar
authentication.password	User password for login when the password authentication mode is used. In the normal mode or keytab login mode, this parameter does not need to be set. The password needs to be encrypted. The encryption method is described as follows: 1. Go to the directory where **encrypt_tool** is located. For example, if the Loader client installation directory is **/opt/hadoopclient/Loader**, run the following command: **cd /opt/hadoopclient/Loader/loader-tools-1.99.3** 2. Run the following command to encrypt the non-encrypted password: **./encrypt_tool** *Unencrypted password* The obtained encrypted password is used as the value of **authentication.password**. **NOTE** If a non-encrypted password contains special characters, the special characters must be escaped. For example, the dollar sign ($) is a special character and can be escaped using single quotation marks ('). If a non-encrypted password contains single quotation marks, use double quotation marks to escape the single quotation marks. If a non-encrypted password contains double quotation marks, use backslashes (\) to escape the double quotation marks. For details, see the shell escape character rules.	-

Configuration parameters	Description	Example Value
use.keytab	Whether to use the keytab mode to log in. • **true** indicates using the keytab file to log in. • **false** indicates using the password to log in.	true
client.principal	User principal for accessing the Loader service when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	loader/hadoop
client.keytab.file	Directory where the used keytab file is located when the keytab authentication mode is used. In the normal mode or password login mode, this parameter does not need to be set.	/opt/client/conf/loader.keytab

Step 4 Run the following command to go to the directory where the **sqoop2-shell** script is located and create a text file in the directory, such as **batchCommand.sh**:

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell

vi batchCommand.sh

An example of **batchCommand.sh** is displayed as follows:

```
View parameters
create connection -c 4 --help

// Create a connector
create connection -c 4 -name oracle-connection --connector-connection-oracleServerIp 10.0.0.1 --connector-connection-oracleServerPort 22 --connector-connection-oracleUser root --connector-connection-oraclePassword xxxxx

Create a job
create job -t import -x 3 --connector-file-inputPath /opt/tempfile --connector-file-fileFilter * --framework-output-outputDirectory /user/loader/1 --framework-output-storageType HBase --framework-throttling-extractorSize 120 --framework-output-fileType TEXT_FILE --connector-file-splitType FILE -name test

Start a job
start job -j 7 -s
```

xxxxx is the password for the connector.

Step 5 Run the following command and the sqoop-shell tool will run the preceding commands in sequence:

./sqoop2-shell batchCommand.sh

The commands above authenticate login by reading configuration files. Alternatively, you can attach the authentication information to the command, that is, use the password mode or Kerberos mode to authenticate login.

Run the following command to authenticate login using the password mode:

./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword* **batchCommand.sh**

Run the following command to authenticate login using the Kerberos mode:

./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal* **batchCommand.sh**

Displaying the **SUCCEEDED** information indicates that the job is started successfully.

```
Welcome to sqoop client
Use the username and password authentication mode
Authentication success.
sqoop:000> create connection -c 4 --help
usage: Show connection viparameters:
   --connector-connection-oraclePassword <arg>
   --connector-connection-oracleServerIp <arg>
   --connector-connection-oracleServerPort <arg>
   --connector-connection-oracleUser <arg>
   --framework-security-maxConnections <arg>
   --name <arg>
===> FINE
sqoop:000> create connection -c 4 -name oracle-connection --connector-connection-oracleServerIp 10.0.0.1
--connector-connection-oracleServerPort 22 --connector-connection-oracleUser root --connector-connection-
oraclePassword xxxxx
Creating connection for connector with id 4
New connection was successfully created with validation status FINE and persistent id 3
===> FINE
sqoop:000> create job -t import -x 3 --connector-file-inputPath /opt/tempfile --connector-file-fileFilter * --
framework-output-outputDirectory /user/loader/1 --framework-output-storageType HDFS --framework-
throttling-extractorSize 120 --framework-output-fileType TEXT_FILE --connector-file-splitType FILE -name
test
Creating job for connection with id 3
New job was successfully created with validation status FINE  and persistent id 7
===> FINE
Submission details
Job ID: 7
Server URL: https://10.0.0.0:21351/loader/
Created by: admintest
Creation date: 2019-12-04 16:37:34 CST
Lastly updated by: admintest
2019-12-04 16:37:34 CST: BOOTING  - Progress is not available
2019-12-04 16:37:42 CST: BOOTING  - 0.00 %
2019-12-04 16:37:42 CST: BOOTING  - 0.00 %
2019-12-04 16:37:57 CST: RUNNING  - 0.00 %
2019-12-04 16:38:12 CST: RUNNING  - 45.00 %
2019-12-04 16:38:12 CST: RUNNING  - 45.00 %
2019-12-04 16:38:27 CST: SUCCEEDED
```

Step 6 In the batch mode, the **-c** parameter can be used to attach a command. sqoop-shell can execute only the attached command at a time.

Run the following command to create a connection:

./sqoop2-shell -c "create connection -c 4 -name oracle-connection --connector-connection-oracleServerIp 10.0.0.1 --connector-connection-oracleServerPort 22 --connector-connection-oracleUser root --connector-connection-oraclePassword xxxxx"

You can also use the password mode or Kerberos mode to attach the authentication information to the command.

Run the following command to authenticate login using the password mode:

./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword* **-c "create connection -c 4 -name oracle-connection --connector-connection-oracleerverIp 10.0.0.1 --connector-connection-oracleServerPort 22 --connector-connection-oracleUser root --connector-connection-oraclePassword xxxxx"**

Run the following command to authenticate login using the Kerberos mode:

./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal* **-c "create connection -c 4 -name oracle-connection --connector-connection-oracleServerIp 10.0.0.1 --connector-connection-oracleServerPort 22 --connector-connection-oracleUser root --connector-connection-oraclePassword xxxxx"**

Displaying the **FINE** information indicates the connection is created successfully.

```
Welcome to sqoop client
Use the username and password authentication mode
Authentication success.
sqoop:000> create connection -c 4 -name oracle-connection --connector-connection-oracleServerIp 10.0.0.1
--connector-connection-oracleServerPort 22 --connector-connection-oracleUser root --connector-connection-
oraclePassword xxxxx
Creating connection for connector with id 4
New connection was successfully created with validation status FINE and persistent id 3
===> FINE
```

----End

20.10 Common Issues About Loader

20.10.1 Data Cannot Be Saved When Loader Jobs Are Configured

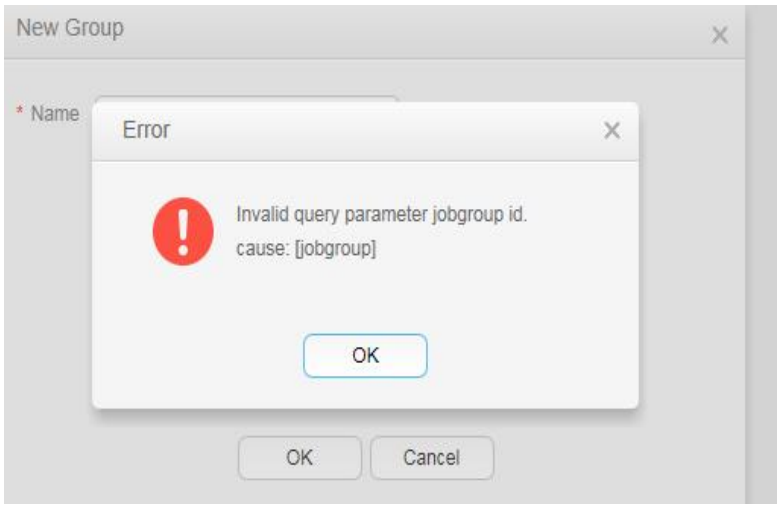
Question

When Internet Explorer 10 or 11 is used to access the Loader page and submit data, an error is reported.

Answer

- Symptom

  After data is saved and submitted, an error similar to "Invalid query parameter jobgroup id. cause: [jobgroup]" is reported.



- Cause

  The POST requests are converted into GET requests after receiving the HTTP 307 response in some Internet Explorer 11 versions. As a result, POST data cannot be delivered to the server.

- Solution

  Use Google Chrome.

20.10.2 Differences Among Connectors Used During the Process of Importing Data from the Oracle Database to HDFS

Question

Three types of connectors are available for importing data from the Oracle database to HDFS using Loader. That is, generic-jdbc-connector, oracle-connector, and oracle-partition-connector. Which one should I select? What are the differences between them?

Answers

- generic-jdbc-connector

  Reads data from the Oracle database in JDBC mode. It is applicable to databases that support JDBC.

  In this mode, data loading performance of Loader is subject to data distribution in a partition column. When data skew occurs (data has only one value or several values) in a partition column, a few Maps process a significant portion of data. As a result, the index becomes invalid, causing a sharp decline in SQL query performance.

  generic-jdbc-connector supports view import and export, but oracle-partition-connector and oracle-connector do not support. Therefore, only this connector can be used to import views.

- Both **oracle-partition-connector** and **oracle-connector**

  can use the ROWID of Oracle for partitioning. oracle-partition-connector is self-developed and oracle-connector is an open-source edition. The two types of connectors share similar performance.

oracle-connector requires more system table permissions. The following lists the read permissions required by the system tables of **oracle-connector** and **oracle-connector**.

- **oracle-connector**: dba_tab_partitions, dba_constraints, dba_tables t, dba_segments, v$version, dba_objects, v$instance, SYS_CONTEXT function, dba_extents, and dba_tab_subpartitions
- **oracle-partition-connector**: DBA_OBJECTS and DBA_EXTENTS

Compared with **generic-jdbc-connector**, **oracle-partition-connector** and **oracle-connector** have the following advantages:

a. Load balancing: Number and scope of data segments are determined by the storage structure (data blocks) of the source table rather than the data on the source table. In terms of granularity, a data block can occupy a partition.
b. Stable performance: Invalid index faults caused by data skew and bound variable snooping can be completely eliminated.
c. Fast query speed: Using data segmentation delivers a higher query speed than that of using index.
d. Excellent horizontal scalability: The number of generated segments increases with the increase of data volume. In this case, ideal performance can be delivered when you increase the number of concurrent tasks. Contrarily, decreasing concurrent tasks saves resources.
e. Simplified data segmentation logic: Problems like precision loss, type compatibility, and bound variables can be prevented.
f. Enhanced usability: Users do not need to create partition columns and tables for Loader.

20.10.3 Why Data Is Not Imported to HDFS After All Data Types of SQL Server Are Selected?

Question

After all data types of SQL Server are selected, data is not imported to HDFS.

```
create table test(rtd1 varchar(20),rtd2 char(20),rtd3 smallint,rtd4 int,rtd5 bigint,rtd6 float,rtd8 decimal(10,3),rtd9 date,rtd10 timestamp,rtd12 binary(20));
insert into test values('ghikg\nhui','sa\tfsd',15,89734,9374293493,14.25,145.22,'2007-12-20',DEFAULT ,1110111);
select * from test;
```

Answer

The data contains the Timestamp data type specific to SQL Server. This data type is irrelevant to time and date and needs to be replaced with the Datetime type.

20.10.4 An Error Is Reported When a Large Amount of Data Is Written to HDFS

Symptom

"NotReplicatedYet Exception: Not replicated yet" is occasionally reported when a large amount of data is written to HDFS.

Figure 20-92 Error message

Answer

The possible causes are as follows:

1. The HDFS client sends a new block application to the NameNode. The NameNode does not process the application in time. As a result, the application times out.
2. The incremental data reporting of DataNodes is too slow. As a result, NameNodes cannot allocate new blocks in a timely manner.

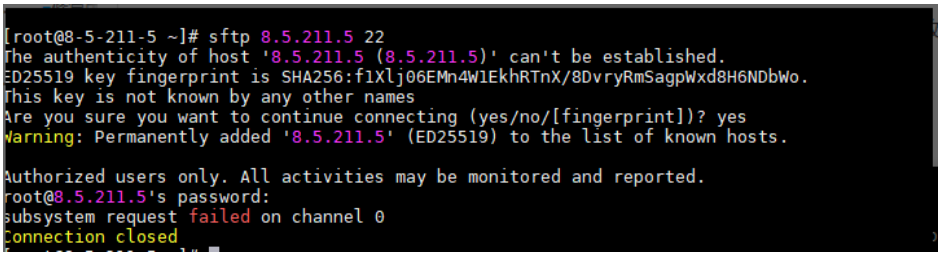
If this error occurs, the job will not become abnormal immediately. The job exception is notified only when the number of retry times exceeds the threshold. Increase the value of the HDFS parameter **dfs.client.block.write.retries**, for example, set the value to **10**.

20.10.5 Failed to Run Jobs Related to the sftp-connector Connector

Symptom

- The jobs related to the sftp-connector connector fail to be executed and "Failed to obtain the SFTP stream" is displayed. *xxx* (Failed to send channel request.)
- Error "subsystem request failed on channel 0. Connection closed" occurs in the SFTP service.

Figure 20-93 SFTP error



Answer

This error occurs because the SFTP service is not enabled.

Procedure

Step 1 Modify the **sshd_config** file, save the modification, and exit.

cd /etc/ssh

vi sshd_config

Subsystem sftp /usr/libexec/openssh/sftp-server

Figure 20-94 Modifying the **sshd_config** file

```
# override default of no subsystems
Subsystem       sftp    /usr/libexec/openssh/sftp-server
```

Step 2 Restart the SFTP service.

systemctl restart sshd.service

----End

21 Using MapReduce

21.1 Configuring the Distributed Cache

Scenarios

Distributed caching is useful in the following scenarios:

Rolling Upgrade

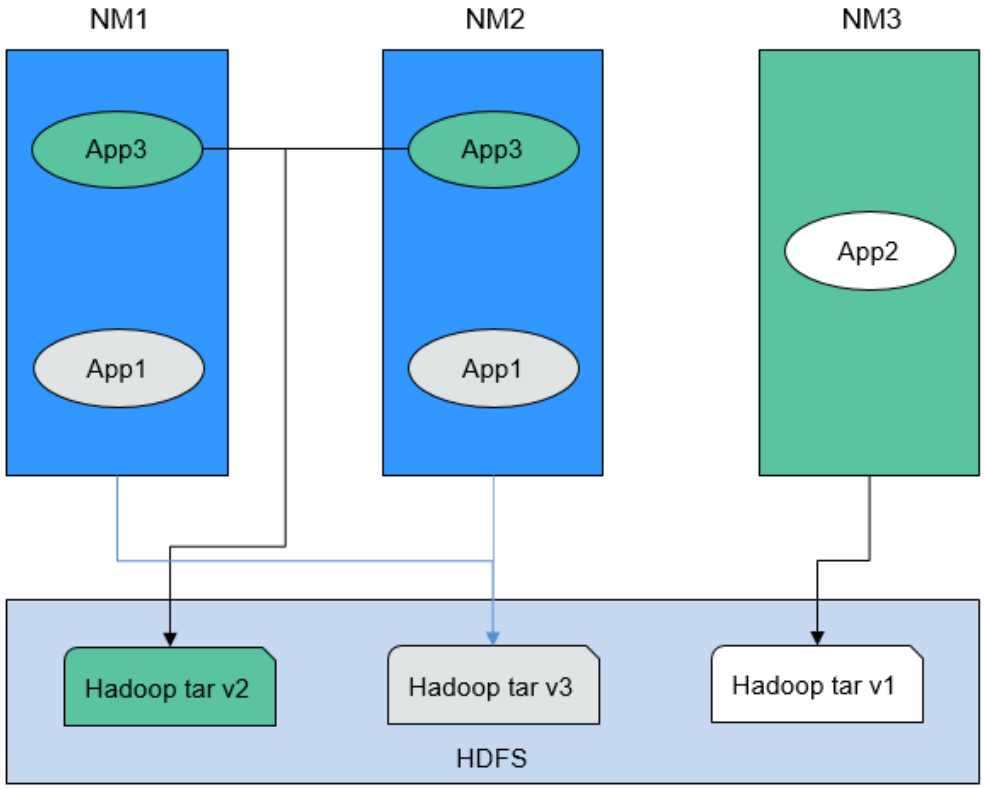
During the upgrade, applications must keep the text content (JAR file or configuration file) unchanged. The content is not based on Yarn of the current version, but on the version when it is submitted. This is a challenging issue. Generally, applications (such as MapReduce, Hive, and Tez) need to be installed locally. Libraries need to be installed on all cluster servers (clients and servers). When a rolling upgrade or downgrade starts in the cluster, the version of the locally installed library changes during application running. During the rolling upgrade, only a few NodeManagers are upgraded first. These NodeManagers obtain the software of the latest version. This leads to inconsistent behavior and can result in run-time errors.

Co-existence of Multiple Yarn Versions

Cluster administrators may run tasks that use multiple versions of Yarn and Hadoop JARs in a cluster. However, this task is difficult to be implemented because the JARs have been localized and have only one version.

The MapReduce application framework can be deployed through the distributed cache and does not depend on the static version copied during installation. Therefore, you can store multiple versions of Hadoop in HDFS and configure the **mapred-site.xml** file to specify the default version used by the task. You can run different versions of MapReduce by setting proper configuration attributes without using the versions deployed in the cluster.

Figure 21-1 Clusters with NodeManagers and Applications of multiple versions



As shown in **Figure 21-1**, the application can use Hadoop JARs in HDFS instead of the local version. Therefore, during the rolling upgrade, even if NodeManager has been upgraded, the application can still run Hadoop of the earlier version.

Configuration Description

Step 1 Save the MapReduce **.tar** package of the specified version to a directory that can be accessed by applications in HDFS, as shown in the following command.

$HADOOP_HOME/**bin/hdfs dfs -put** *hadoop-x.tar.gz /mapred/framework/*

Step 2 Set parameters in the *client installation path*/**Yarn/config/mapred-site.xml** file based on **Table 21-1**.

Table 21-1 Distributed cache parameters

Parameter	Description	Default Value
mapreduce.application.framework.path	Indicates the URL directing to the archive location. **NOTE** This property can also create an alias for the archive if the URL fragment identity name is specified as follows. In this example, the alias is set to **mr-framework**. <property> <name>mapreduce.application.framework.path</name> <value>hdfs:/mapred/framework/hadoop-x.tar.gz#mr-framework</value> </property>	NA

Parameter	Description	Default Value
mapreduce.application.classpath	Indicates the parameter property, which contains the MapReduce JARs in the class directory. **NOTE** For example, the alias **mr-framework** used in the framework path is used to match the directory. <property> <name>mapreduce.application.classpath</name> <value>$PWD/mr-framework/hadoop/share/hadoop/mapreduce/*:$PWD/mr-framework/hadoop/share/hadoop/mapreduce/lib/*:$PWD/mr-framework/hadoop/share/hadoop/common/*:$PWD/mr-framework/hadoop/share/hadoop/common/lib/*:$PWD/mr-framework/hadoop/share/hadoop/yarn/*:$PWD/mr-framework/hadoop/share/hadoop/yarn/lib/*:$PWD/mr-framework/hadoop/share/hadoop/hdfs/*:$PWD/mr-framework/hadoop/share/hadoop/hdfs/lib/*:/etc/hadoop/conf/secure</value></property>	N/A

You can upload MapReduce tarballs of multiple versions to HDFS. Different **mapred-site.xml** files indicate different locations. After that, you can run tasks for a specific **mapred-site.xml** file. The following is an example of running an MapReduce task for the MapReduce tarball of the *x* version:

hadoop jar *share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar pi* **-conf etc/hadoop-x/mapred-site.xml** *10 10*

----End

21.2 Configuring the MapReduce Shuffle Address

Scenario

When the MapReduce shuffle service is started, it attempts to bind an IP address based on local host. If the MapReduce shuffle service is required to connect to a specific IP address, no configuration is available. The following description allows you to configure a connection to a specific IP address.

Configuration

To bind a specific IP address to the MapReduce shuffle service, set the following parameters in the **mapred-site.xml** configuration file (For example, the path is **${BIGDATA_HOME}/FusionInsight_HD_*xxx*/*x_x*x_NodeManager/etc/mapred-site.xml**.) of the node where the NodeManager instance resides:

Table 21-2 Parameter description

Parameter	Description	Default Value
mapreduce.shuffle.address	Indicates the specified address to run the shuffle service. The format is *IP:PORT*. The default value is empty. If this parameter is left empty, the local host IP address is bound. The default port number is 13562. **NOTE** If the value of *PORT* is different from that of **mapreduce.shuffle.port**, the **mapreduce.shuffle.port** value does not take effect.	-

21.3 Configuring the MapReduce Cluster Administrator List

Scenario

This function is used to specify the MapReduce cluster administrator.

The cluster administrator list is specified by **mapreduce.cluster.administrators**. The cluster administrator **admin** has all operation permissions.

Configuration

On the **All Configurations** page of the MapReduce service, enter a parameter name in the search box. For details, see **Modifying Cluster Service Configuration Parameters**.

Table 21-3 Parameter description

Parameter	Description	Default Value
mapreduce.cluster.acls.enabled	Indicates whether to enable permission control on Job History Server.	true

Parameter	Description	Default Value
mapreduce.cluster.administrators	Indicates the administrator list of the MapReduce cluster. You can configure both users and user groups. Multiple users or user groups are separated by commas (,), and users and user groups are separated by spaces, for example, userA,userB groupA,groupB. The value * indicates all users or user groups.	mapred supergroup,System_administrator_186

21.4 Transmitting MapReduce Tasks from Windows to Linux

Scenarios

If you want to transmit a job from Windows to Linux, set **mapreduce.app-submission.cross-platform** to **true**. If this parameter is unavailable for a cluster or its value is **false**, the function of transmitting MapReduce tasks from Windows to Linux is not supported. In this case, perform the following operations to add this parameter or change its value to enable this function:

Configuration Description

Adjust the following parameter in the **mapred-site.xml** configuration file on the client to enable the running of MapReduce tasks: The **mapred-site.xml** configuration file is in the **config** directory of the client installation path, for example, **/opt/client/Yarn/config**.

Table 21-4 Parameters

Parameter	Description	Default Value
mapreduce.app-submission.cross-platform	Indicates whether to support running of MapReduce tasks after they are transmitted from Windows to Linux. When the parameter value is **true**, the running of MapReduce tasks is supported. When the parameter value is **false**, the running of MapReduce tasks is not supported.	true

21.5 Configuring the Archiving and Clearing Mechanism for MapReduce Task Logs

Scenario

Job and task logs are generated during execution of a MapReduce application.

- Job logs are generated by the MRApplicationMaster, which record details about the start and running time of jobs and each task, Counter value, and other information. After being analyzed by HistoryServer, the job logs are used to view job execution details.
- A task log records the log information generated by each task running in a container. By default, task logs are stored only on the local disk of each NodeManager. After the log aggregation function is enabled, the NodeManager merges local task logs and writes them into HDFS after job execution completes.

The job logs and task logs of the MapReduce are stored on HDFS (when the log aggregation function is enabled). If the mechanism for periodically archiving and deleting log files is not configured for a cluster with a large number of computation tasks, the log files will occupy large memory space of HDFS and increase the cluster load.

Log archive is implemented by Hadoop Archives. The number (number of Map tasks) of concurrent archiving tasks started by the Hadoop Archives is related to the total size of log files to be archived. The formula is as follows: Number of concurrent archive tasks = Total size of log files to be archived/Size of archive files.

Configuration

Go to the **All Configurations** page of the MapReduce service. For details, see **Modifying Cluster Service Configuration Parameters**.

Enter the parameter name in the search box, change the parameter value, and save the configuration. On the **Dashboard** tab page of the Mapreduce service, choose **More** > **Synchronize Configuration**. After the synchronization is complete, restart the Mapreduce service.

- Job log parameters:

  Table 21-5 Parameter description

Parameter	Description	Default Value
mapreduce.jobhistory.cleaner.enable	Whether to enable the job log file deletion function.	true

Parameter	Description	Default Value
mapreduce.jobhistory.cleaner.interval-ms	Period for starting a log file cleanup. Only log files whose retention period is longer than the time specified by **mapreduce.jobhistory.max-age-ms** can be deleted.	86,400,000 ms (1 day)
mapreduce.jobhistory.max-age-ms	Log files whose retention period is longer than the retention period in milliseconds specified by this parameter will be deleted.	1,296,000,000 ms (15 days)

- Task log parameters:

  Table 21-6 Parameter description

Parameter	Description	Default Value
yarn.log-aggregation.archive.files.minimum	Indicates the minimum number of archived MapReduce job log files. The archiving task starts when the number of files in the **yarn.nodemanager.remote-app-log-dir** folder is greater than or equal to the value of this parameter.	5,000
yarn.log-aggregation.archive-check-interval-seconds	Indicates the MapReduce job log archiving interval, in seconds. Log files are archived only when the number of log files reaches the value of **yarn.log-aggregation.archive.files.minimum**. The archiving function is disabled when the period is set to **0** or **-1**.	-1
yarn.log-aggregation.retain-seconds	Indicates the retention period on HDFS for archiving the MapReduce job logs. The value **-1** indicates that log files are stored permanently.	1,296,000
yarn.log-aggregation.retain-check-interval-seconds	Indicates the check period (in seconds) of the MapReduce job log deletion task. If this parameter is set to **-1**, the check period is one tenth of the log retention period.	86400

NOTE

If task logs occupy too much HDFS storage space, modify the **mapreduce.jobhistory.max-age-ms** and **yarn.log-aggregation.retain-check-interval-seconds** configuration items to control the storage duration of task logs.

21.6 MapReduce Performance Tuning

21.6.1 MapReduce Optimization Configuration for Multiple CPU Cores

Scenario

Optimization can be performed when the number of CPU cores is large, for example, the number of CPU cores is three times the number of disks.

Procedure

You can set the following parameters in either of the following ways:

- Configuration on the server:

  On the **All Configurations** page of the Yarn service, enter a parameter name in the search box. For details, see **Modifying Cluster Service Configuration Parameters**.

- Configuration on the client:

  Modify the corresponding configuration file on the client.

  NOTE

  - Path of configuration files on the HDFS client: *Client installation directory*/**HDFS/hadoop/etc/hadoop/hdfs-site.xml**
  - Path of configuration files on the Yarn client: *Client installation directory*/**HDFS/hadoop/etc/hadoop/yarn-site.xml**.
  - Path of configuration files on the MapReduce client: *Client installation directory*/**HDFS/hadoop/etc/hadoop/mapred-site.xml**.

Table 21-7 Settings of multiple CPU cores

Configuration	Description	Parameter	Default Value	Server/ Client	Impact	Remarks
Number of slots in a node container	The combination of the following parameters determines the number of concurrent tasks (Map and Reduce tasks) of each node: • yarn.nodemanager.resource.memory-mb • mapreduce.map.memory.mb • mapreduce.reduce.memory.mb	yarn.nodemanager.resource.memory-mb **NOTE** You need to configure this parameter on FusionInsight Manager.	16384	Server	If data needs to be read from and written into disks for all tasks (Map/ Reduce tasks), a disk may be accessed by multiple processes at the same time, which leads to poor disk I/O performance. To ensure disk I/O performance, the number of concurrent access requests from a client to a disk cannot exceed 3.	The maximum number of concurrent containers must be [2.5 x Number of disks configured in Hadoop].
		mapreduce.map.memory.mb **NOTE** You need to set this parameter in the configuration file on the client in the *Client installation directory*/**HDFS/hadoop/etc/hadoop/mapred-site.xml** path.	4096	Client		
		mapreduce.reduce.memory.mb **NOTE** You need to set this parameter in the configuration file on the client in the *Client installation directory*/**HDFS/hadoop/etc/hadoop/mapred-site.xml** path.	4096	Client		

Configuration	Description	Parameter	Default Value	Server/ Client	Impact	Remarks
Map output and compression	The Map task output before being written into disks can be compressed. This can save disk space, offer faster data write, and reduce the data traffic delivered to Reducer. You need to configure the following parameters on the client: • **mapreduce.map.output.compress**: The Map task output can be compressed before it is transmitted over the network. It is a per-job	mapreduce.map.output.compress **NOTE** You need to set this parameter in the configuration file on the client in the *Client installation directory*/**HDFS/hadoop/etc/hadoop/mapred-site.xml** path.	true	Client	The disk I/O is the bottleneck. Therefore, use a compression algorithm with a high compression rate.	Snappy is used. The benchmark test results show that Snappy delivers high performance and efficiency.
		mapreduce.map.output.compress.codec **NOTE** You need to set this parameter in the configuration file on the client in the *Client installation directory*/**HDFS/hadoop/etc/hadoop/mapred-site.xml** path.	org.apache.hadoop.io.compress.Lz4Codec	Client		

Configuration	Description	Parameter	Default Value	Server/ Client	Impact	Remarks
	configuration. • **mapreduce.map.output.compress.codec**: the codec used for data compression					
Spills	mapreduce.map.sort.spill.percent	mapreduce.map.sort.spill.percent **NOTE** You need to set this parameter in the configuration file on the client in the *Client installation directory*/**HDFS/hadoop/etc/hadoop/mapred-site.xml** path.	0.8	Client	Disk I/Os are the bottleneck. You can set the value of **mapreduce.task.io.sort.mb** to minimize the memory spilled to the disk.	-

Configuration	Description	Parameter	Default Value	Server/ Client	Impact	Remarks
Data packet size	When the HDFS client writes data to a data node, the data will be accumulated until a packet is generated. Then, the packet is transmitted over the network. **dfs.client-write-packet-size** specifies the data packet size. It can be specified by each job.	dfs.client-write-packet-size **NOTE** You need to set this parameter in the configuration file on the client in the *Client installation directory*/**HDFS/hadoop/etc/hadoop/hdfs-site.xml/** path.	262144	Client	The data node receives data packets from the HDFS client and writes data into disks through single threads. When disks are in the concurrent write state, increasing the data packet size can reduce the disk seek time and improve the I/O performance.	dfs.client-write-packet-size = 262144

21.6.2 Determining the Job Baseline

Scenario

The performance optimization effect is verified by comparing actual values with the baseline data. Therefore, determining optimal job baseline is critical to performance optimization.

When determining the job baseline, comply with the following rules:

- Making full use of cluster resources
- Setting the number of Map and Reduce tasks appropriately
- Setting the runtime of each task appropriately

Procedure

- **Rule 1: Making full use of cluster resources**

  Enable all nodes to handle tasks as actively as they can when a job is executed. Maximizing the number of concurrent tasks helps make full use of resources. You can achieve this purpose by adjusting the data volume to be processed and the number of Map and Reduce tasks.

  You can set **mapreduce.job.reduces** to control the number of Reduce tasks.

  The number of Map tasks depends on the InputFormat type and whether the data file to be processed can be split. By default, TextFileInputFormat allocates Map tasks based on the number of blocks, that is, one Map task for each block. You can adjust the following parameters to improve resource utilization.

  Parameter portal:

  On the **All Configurations** page of the Yarn service, enter a parameter name in the search box. For details, see **Modifying Cluster Service Configuration Parameters**.

Parameter	Description	Default Value
mapreduce.input.fileinputformat.split.maxsize	Indicates the maximum size of the data block into which the Map input information is to be split. The shard size can be calculated based on its size customized by the user and the block size of each file. The formula is as follows: `splitSize = Math.max(minSize, Math.min(maxSize, blockSize))` If **maxSize** is bigger than **blockSize**, a block is a shard. If **maxSize** is smaller than **blockSize**, a block will be split into multiple shards. If the size of the remaining data in a block is smaller than **splitSize**, the remaining data will be treated as a separated shard.	-
mapreduce.input.fileinputformat.split.minsize	Indicates the minimum size of a data shard.	0

- **Principle 2: Setting Reduce tasks to be executed in one round.**

  Avoid the following scenarios:

  – Most of Reduce tasks are completed in the first round, but there is still one Reduce task left running. The execution of the last Reduce task extends the runtime of the job. Therefore, reduce the number of Reduce tasks to enable all of them to run at the same time.

  - All Map tasks are completed, but there are still Reduce tasks running on some nodes. In this case, the cluster resources are not fully utilized. You need to increase the number of Reduce tasks to enable each node to handle tasks.

- **Rule 3: Setting the runtime of each task appropriately**

  If each Map or Reduce task of a job takes only a few seconds, most time of the job is wasted on scheduling tasks and starting and stopping processes. Therefore, you need to increase the data volume to be processed in each task. The preferred processing time for each task is 1 minute.

  You can configure the following parameters to adjust the processing time in a task.

  Parameter portal:

  On the **All Configurations** page of the Yarn service, enter a parameter name in the search box. For details, see **Modifying Cluster Service Configuration Parameters**.

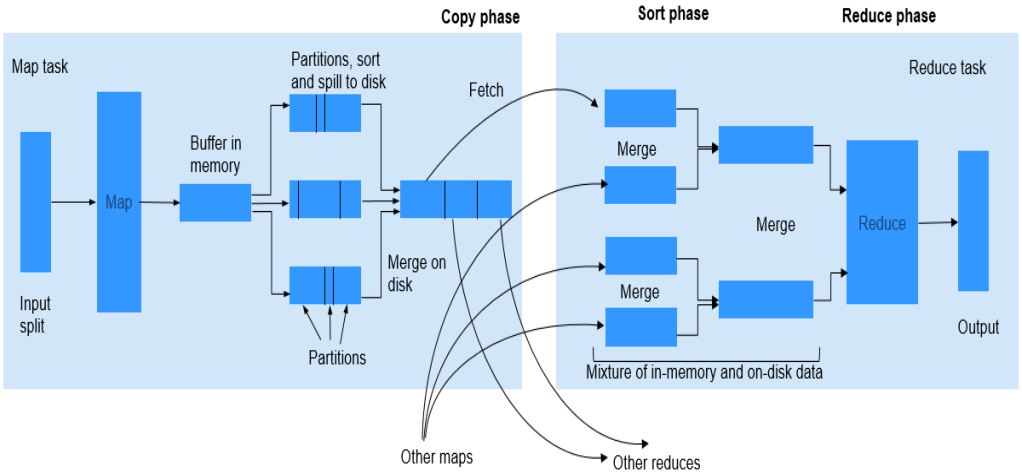
Parameter	Description	Default Value
mapreduce.input.fileinputformat.split.maxsize	Indicates the maximum size of the data block into which the Map input information is to be split. The shard size can be calculated based on its size customized by the user and the block size of each file. The formula is as follows: `splitSize = Math.max(minSize, Math.min(maxSize, blockSize))` If **maxSize** is bigger than **blockSize**, a block is a shard. If **maxSize** is smaller than **blockSize**, a block will be split into multiple shards. If the size of the remaining data in a block is smaller than **splitSize**, the remaining data will be treated as a separated shard.	-
mapreduce.input.fileinputformat.split.minsize	Indicates the minimum size of a data shard.	0

21.6.3 MapReduce Shuffle Tuning

Scenario

During the shuffle procedure of MapReduce, the Map task writes intermediate data into disks, and the Reduce task copies and adds the data to the reduce function. Hadoop provides lots of parameters for the optimization.

Figure 21-2 Shuffle process



Procedure

1. **Improving Performance in Map Phase**
   - Determine the memory used by Map.

     To determine whether Map has sufficient memory, check the number of GCs and the ratio of the GC time over the total task time in counters of completed jobs. Normally, the GC time cannot exceed 10% of the task time (that is, GC time elapsed (ms)/CPU time spent (ms) < 10%).

     You can improve Map performance by adjusting the following parameters.

     Parameter portal:

     On the **All Configurations** page of the Yarn service, enter a parameter name in the search box. For details, see **Modifying Cluster Service Configuration Parameters**.

     It is recommended that the **-Xmx** in **mapreduce.map.java.opts** is 0.8 times the value of **mapreduce.map.memory.mb**.

     Table 21-8 Parameter description

Parameter	Description	Default Value
mapreduce.map.memory.mb	Memory restriction of a Map task.	4096

Parameter	Description	Default Value
mapreduce.map.java.opts	JVM parameter of the Map subtask. If this parameter is set, it will replace the **mapred.child.java.opts** parameter. If **-Xmx** is not set, the value of **Xmx** is calculated based on **mapreduce.map.memory.mb** and **mapreduce.job.heap.memory-mb.ratio**.	• Clusters with Kerberos authentication enabled: -Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false -Djava.security.krb5.conf=*${BIGDATA_HOME}*/common/runtime/krb5.conf -Dbeetle.application.home.path=*${BIGDATA_HOME}*/common/runtime/security/config • Clusters with Kerberos authentication disabled: -Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false -Dbeetle.application.home.path=*${BIGDATA_HOME}*/common/runtime/security/config

– Using Combiner

  Combiner is an optional procedure in the Map phase, in which the intermediate results with the same key value are combined. Generally, set the reduce class to combiner. Combiner helps reduce the intermediate result output of Map, thereby consuming less network bandwidth during the shuffle process. You can use the following API to set a combiner class for a specific job.

Table 21-9 Combiner API

Class	API	Description
org.apache.hadoop.mapreduce.Job	public void setCombinerClass(Class<? extends Reducer> cls)	API used to set a combiner class for a specific job.

2. **Improving Performance in Copy Phase**

   Determine to compress data or not.

   Compress the intermediate output of Map. Data compression reduces the data to be transferred over the network. However, data compression and decompression consume more CPU. Determine whether to compress the intermediate results of Map based on site requirements. If a task is bandwidth-intensive, data compression improves processing performance. As for the bulkload optimization, compression of the intermediate output improves the performance by 60%.

   To improve copy performance, set **mapreduce.map.output.compress** to **true** and **mapreduce.map.output.compress.codec** to **org.apache.hadoop.io.compress.SnappyCodec**.

3. **Improving Performance in Merge Phase**

   To improve merge performance, configure the following parameters to reduce the number of times that Reduce writes data to disks.

   Parameter portal:

   On the **All Configurations** page of the Yarn service, enter a parameter name in the search box. For details, see **Modifying Cluster Service Configuration Parameters**.

   Table 21-10 Parameter description

Parameter	Description	Default Value
mapreduce.reduce.merge.inmem.threshold	Threshold of the number of files for the in-memory merge process. When the accumulated number of files reaches the threshold, the process of in-memory merge and spilling to disks is initiated. If the value is less than or equal to **0**, the threshold does not take effect and the merge is triggered only based on the RAMFS memory usage.	1000

Parameter	Description	Default Value
mapreduce.reduce.shuffle.merge.percent	Usage threshold for initiating in-memory merge, indicating the percentage of memory allocated to the Map outputs (defined by **mapreduce.reduce.shuffle.input.buffer.percent**).	0.66
mapreduce.reduce.shuffle.input.buffer.percent	Percentage of memory to be allocated from the maximum heap size to storing Map outputs during the Shuffle.	0.70
mapreduce.reduce.input.buffer.percent	Percentage of memory (relative to the maximum heap size) to retain Map outputs during the Reduce. When the Shuffle is completed, all remaining Map outputs in memory must use less than this threshold before the Reduce begins.	0.0

21.6.4 AM Optimization for Big MapReduce Tasks

Scenario

A big task containing 100,000 Map tasks fails. The query result shows that ApplicationMaster (AM) responds slowly and finally times out.

When the number of tasks increases, the number of objects managed by the AM increases, which requires much more memory for management. The default memory heap for AM is 1 GB.

Procedure

You can improve the AM performance by setting the following parameters.

Navigation path for setting parameters:

Adjust the following parameters in the **mapred-site.xml** configuration file on the client to adjust the following parameters: The **mapred-site.xml** configuration file is in the **conf** directory of the client installation path, for example, **/opt/client/Yarn/config**.

Parameter	Description	Default Value
yarn.app.mapreduce.am.resource.mb	This parameter must be greater than the heap size specified by **yarn.app.mapreduce.am.command-opts**. Unit: MB	1536
yarn.app.mapreduce.am.command-opts	Indicates the JVM startup parameters loaded to MapReduce ApplicationMaster.	-Xmx1024m -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -verbose:gc -Djava.security.krb5.conf=${KRB5_CONFIG} -Dhadoop.home.dir=*${BIGDATA_HOME}*/FusionInsight_HD_*xxx*/install/FusionInsight-Hadoop-*xxx*/hadoop

21.6.5 Speculative Execution

Scenario

If a cluster has hundreds or thousands of nodes, the hardware or software fault of a node may prolong the execution time of the entire task (as most tasks are already completed, the system is still waiting for the task running on the faulty node). Speculative execution allows a task to be executed on multiple machines. You can disable speculative execution for small clusters.

Procedure

Navigation path for setting parameters:

On the **All Configurations** page of the Yarn service, enter a parameter name in the search box. For details, see **Modifying Cluster Service Configuration Parameters**.

Parameter	Description	Default Value
mapreduce.map.speculative	Sets whether to execute multiple instances of some map tasks concurrently. **true** indicates that speculative execution is enabled.	false
mapreduce.reduce.speculative	Sets whether to execute multiple instances of some reduce tasks concurrently. **true** indicates that speculative execution is enabled.	false

21.6.6 Using Slow Start

Scenario

The Slow Start feature specifies the proportion of Map tasks to be completed before Reduce tasks are started. If the Reduce tasks are started too early, resources will be occupied, thereby reducing task running efficiency. However, if the Reduce tasks are started at an appropriate time, resource usage during shuffle and task running efficiency will be improved. For example, the MapReduce job includes 15 Map tasks and a cluster can start 10 Map tasks, there are 5 Map tasks remained after a round of Map tasks is completed and the cluster has available resources. In this case, you can configure the value of Slow Start to a value less than 1 (for example, 0.8), then the Reduce tasks can make use of the remaining cluster resources.

Procedure

Parameter portal:

On the **All Configurations** page of the MapReduce service, enter a parameter name in the search box. For details, see **Modifying Cluster Service Configuration Parameters**.

Parameter	Description	Default Value
mapreduce.job.reduce.slowstart.completedmaps	Fraction of the number of Maps in the job which should be completed before Reduces are scheduled for the job. By default, the Reduce tasks start when all the Map tasks are completed.	1.0

21.6.7 Optimizing Performance for Committing MR Jobs

Scenario

By default, if an MR job generates a large number of output files, it takes a long time for the job to commit the temporary outputs of a task to the final output directory in the commit phase. In large clusters, the time-consuming commit process of jobs greatly affects the performance.

In this case, you can set the **mapreduce.fileoutputcommitter.algorithm.version** to **2** to improve the performance in the commit phase of MR jobs.

Procedure

Navigation path for setting parameters:

On the **All Configurations** page of the Yarn service, enter a parameter name in the search box. For details, see **Modifying Cluster Service Configuration Parameters**.

Table 21-11 Parameter description

Parameter	Description	Default Value
mapreduce.fileoutputcommitter.algorithm.version	Indicates the algorithm version submitted by a job. The value is **1** or **2**. **NOTE** **2** is the recommended algorithm version. This algorithm enables tasks to directly commit the output results of each task to the final result output directory, reducing the time for the results of large jobs are committed.	2

21.6.8 Reducing Client Application Failure Rate

Scenario

When the network is unstable or the cluster I/O and CPU are overloaded, client applications might encounter running failures.

Configuration

Adjust the following parameters in the **mapred-site.xml** configuration file on the client to reduce the client application failure rate:

NOTE

The **mapred-site.xml** configuration file is in the **conf** directory of the client installation path, for example, **/opt/client/Yarn/config**.

Table 21-12 Parameter description

Parameter	Description	Default Value
mapreduce.reduce.shuffle.max-host-failures	Indicates the number of allowed failures of an MR task to read remote shuffle data in the Reduce process. When the number is set to be over 5, the client application failure rate can be reduced.	5
mapreduce.client.submit.file.replication	Indicates the backup of job files on HDFS. MR tasks are dependent on the job files during running. When the number of backups is set to be over 10, the client application failure rate can be reduced.	10

21.7 Mapreduce Log Overview

Log Description

Log paths:

- JobhistoryServer: **/var/log/Bigdata/mapreduce/jobhistory** (run log) and **/var/log/Bigdata/audit/mapreduce/jobhistory** (audit log)
- Container: **/srv/BigData/hadoop/data1/nm/containerlogs/application_${appid}/container_{$contid}**

NOTE

The logs of running tasks are stored in the preceding paths. After the running is complete, the system determines whether to aggregate the logs to an HDFS directory based on the YARN configuration. For details, see **YARN Common Configuration Parameters**.

Log archive rule:

The automatic compression and archive function is enabled for MapReduce logs. By default, a log file is automatically compressed when the size of the log file is greater than 50 MB. The name of the compressed log file is in the following format: *<Name of the original log>-<yyyy-mm-dd_hh-mm-ss>.[NO.]***.log.zip**. A maximum of 100 latest compressed files are reserved. The number of compressed files can be configured on the parameter configuration page.

In MapReduce, JobhistoryServer cleans the old log files stored in HDFS periodically. The default storage directory is **/mr-history/done**. **mapreduce.jobhistory.max-age-ms** is used to set the cleanup interval. The default value of this parameter is 1,296,000,000 ms, which indicates 15 days.

Table 21-13 MapReduce log list

Type	Name	Description
Run log	jhs-daemon-start-stop.log	Startup log file of the daemon process
	hadoop-<SSH_USER>-jhshadaemon-<hostname>.log	Run log file of the daemon process
	hadoop-<SSH_USER>-<process_name>-<hostname>.out	Log that records the MapReduce running environment information
	historyserver-<SSH_USER>-<DATE>-<PID>-gc.log	Log that records the garbage collection of the MapReduce service
	jhs-haCheck.log	Log that records the active and standby status of MapReduce instances

Type	Name	Description
	yarn-start-stop.log	Log that records the startup and stop of the MapReduce service
	yarn-prestart.log	Log that records cluster operations before the MapReduce service startup
	yarn-postinstall.log	Work log before the MapReduce service startup and after the installation
	yarn-cleanup.log	Log that records the cleanup logs about the uninstallation of the MapReduce service
	mapred-service-check.log	Log that records the health check details of the MapReduce service
	container_{$contid}	Container log
	hadoop-<SSH_USER>-<process_name>-<hostname>.log	MR run log
	mapred-switch-jhs.log	MR active/standby switchover log
	env.log	Environment information log before the instance is started or stopped
Audit log	mapred-audit-jobhistory.log	MapReduce operation audit log
	SecurityAuth.audit	MapReduce security audit log

Log Level

Table 21-14 describes the log levels supported by MapReduce The log levels are FATAL, ERROR, WARN, INFO, and DEBUG from high priority to low. Logs whose levels are higher than or equal to the specified level are printed. The number of printed logs decreases as the specified log level increases.

Table 21-14 Log level

Level	Description
FATAL	Logs of this level record critical error information about the current event processing.
ERROR	Logs of this level record error information about the current event processing.
WARN	Logs of this level record unexpected alarm information about the current event processing.
INFO	Logs of this level record normal running status information about the system and events.
DEBUG	Logs of this level record the system information and system debugging information.

To modify log levels, perform the following operations:

Step 1 Go to the **All Configurations** page of the MapReduce service. For details, see **Modifying Cluster Service Configuration Parameters**.

Step 2 On the left menu bar, select the log menu of the target role.

Step 3 Select a desired log level.

Step 4 Save the configuration. In the displayed dialog box, click **OK** to make the configurations take effect.

NOTE

The configurations take effect immediately without restarting the service.

----End

Log Format

The following table lists the MapReduce log formats.

Table 21-15 Log format

Type	Format	Example
Run log	<*yyyy-MM-dd HH:mm:ss,SSS*>|<*Log level*>|<*Name of the thread that generates the log*>|<*Message in the log*>|<*Location where the log event occurs*>	2020-01-26 14:18:59,109 | INFO | main | Client environment:java.compiler=<NA> | org.apache.zookeeper.Environment.logEnv(Environment.java:100)

Type	Format	Example
Audit log	<*yyyy-MM-dd HH:mm:ss,SSS*>|<*Log level*>|<*Name of the thread that generates the log*>|<*Message in the log*>|<*Location where the log event occurs*>	2020-01-26 14:24:43,605 | INFO | main-EventThread | USER=omm OPERATION=refreshAdminAcls TARGET=AdminService RESULT=SUCCESS | org.apache.hadoop.yarn.server.resourcemanager.RMAuditLogger$LogLevel $6.printLog(RMAuditLogger.java:91)

21.8 Common Issues About MapReduce

21.8.1 After an Active/Standby Switchover of ResourceManager Occurs, a Task Is Interrupted and Runs for a Long Time

Question

During the running of a MapReduce task, active/standby switchover of ResourceManager occurs. After the switchover is complete, the MapReduce task continues to execute, but runs for an excessively long time.

Answer

The ResourceManager HA function has been enabled, but the Work-preserving RM restart function is not enabled.

If the Work-preserving RM restart function is not enabled, the container will be killed during the ResourceManager switchover. As a result, Application Master times out. For details about the Work-preserving RM restart function, visit the following website:

Versions earlier than MRS 3.2.0: **http://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-site/ResourceManagerRestart.html**

MRS 3.2.0 or later: **https://hadoop.apache.org/docs/r3.3.1/hadoop-yarn/hadoop-yarn-site/ResourceManagerRestart.html**

To resolve this issue, perform the following operation:

Set the **yarn.resourcemanager.work-preserving-recovery.enabled** parameter to **true** to enable the Work-preserving RM restart function.

yarn.resourcemanager.work-preserving-recovery.enabled=**true**

21.8.2 Why Does a MapReduce Task Stay Unchanged for a Long Time?

Question

MapReduce job is not progressing for long time

Answer

This is because of less memory. When the memory is less, the time taken by the job to copy the map output increases significantly.

In order to reduce the waiting time, increase the heap memory.

The task configuration can be optimized based on the number of mappers and the data size of each mapper. Optimize the following parameters in the *client installation path*/**Yarn/config/mapred-site.xml** file based on the size of the input data:

- **mapreduce.reduce.memory.mb**
- **mapreduce.reduce.java.opts**

Example: If the data size is 5 GB with 10 mappers, then the ideal heap memory would be 1.5 GB. Increase the heap memory size according with the increase in data size.

21.8.3 Why the Client Hangs During Job Running?

Question

Why is the client unavailable when the MR ApplicationMaster or ResourceManager is moved to the D state during job running?

Answer

When a task is running, the MR ApplicationMaster or ResourceManager is moved to D state (uninterrupted sleep state) or T state (stopped state). The client waits to return the task running state, but the MR ApplicationMaster does not return. Therefore, the client remains in the waiting state.

To avoid the preceding scenario, use the **ipc.client.rpc.timeout** configuration item in the **core-site.xml** file to set the client timeout interval.

The value of this parameter is millisecond. The default value is **0**, indicating that no timeout occurs. The client timeout interval ranges from 0 ms to 2,147,483,647 ms.

NOTE

- If the Hadoop process is in the D state, restart the node where the process is located.
- The **core-site.xml** configuration file is stored in the **conf** directory of the client installation path, for example, **/opt/client/Yarn/config**.

21.8.4 Why Cannot HDFS_DELEGATION_TOKEN Be Found in the Cache?

Question

In security mode, why delegation token HDFS_DELEGATION_TOKEN is not found in the cache?

Answer

In MapReduce, by default HDFS_DELEGATION_TOKEN will be canceled after the job completion. So if the token has to be re- used for the next job then the token will not be found in the cache.

To re-use the same token in subsequent job set the below parameter for the MR job configuration. When it is false the user can re-sue the same token.

```
jobConf.setBoolean("mapreduce.job.complete.cancel.delegation.tokens", false);
```

21.8.5 How Do I Set the Task Priority When Submitting a MapReduce Task?

Question

How do I set the job priority when submitting a MapReduce task?

Answer

You can add the parameter **-Dmapreduce.job.priority=<priority>** in the command to set task priority when submitting MapReduce tasks on the client. The format is as follows:

yarn jar *<jar> [mainClass]* ***-Dmapreduce.job.priority=****<priority> [path1] [path2]*

The parameters in the command are described as follows:

- *<jar>*: specifies the name of the JAR package to be run.
- *[mainClass]*: specifies the **main** method of the class for an application project in a JAR file.
- *<priority>*: specifies the priority of a task. The value can be **VERY_HIGH**, **HIGH**, **NORMAL**, **LOW**, or **VERY_LOW**.
- *[path1]*: specifies the data input path.
- *[path2]*: specifies the data output path.

For example, set the **/opt/client/HDFS/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples*.jar** file to a high-priority task.

yarn jar /opt/client/HDFS/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples*.jar wordcount -Dmapreduce.job.priority=VERY_HIGH /DATA.txt /out/

21.8.6 Why Physical Memory Overflow Occurs If a MapReduce Task Fails?

Question

The HBase bulkload task has 210,000 Map tasks and 10,000 Reduce tasks. The MapReduce task fails to be executed, and the physical memory of ApplicationMaster overflows.

```
For more detailed output, check the application tracking page:https://bigdata-55:8090/cluster/app/
application_1449841777199_0003
Then click on links to logs of each attempt.
Diagnostics: Container [pid=21557,containerID=container_1449841777199_0003_02_000001]  is running
beyond physical memory limits
Current usage: 1.0 GB of 1 GB physical memory used; 3.6 GB of 5 GB virtual memory used. Killing container.
Dump of the process-tree for container_1449841777199_0003_02_000001 :
|- PID PPID PGRPID SESSID CMD_NAME USER_MODE_TIME(MILLIS) SYSTEM_TIME(MILLIS)
VMEM_USAGE(BYTES) RSSMEM_USAGE(PAGES) FULL_CMD_LINE
|- 21584 21557 21557 21557 (java) 12342 1627 3871748096 271331 ${BIGDATA_HOME}/jdk1.8.0_51//bin/
java
-Djava.io.tmpdir=/srv/BigData/hadoop/data1/nm/localdir/usercache/hbase/appcache/
application_1449841777199_0003/container_1449841777199_0003_02_000001/tmp -
Dlog4j.configuration=container-log4j.properties
-Dyarn.app.container.log.dir=/srv/BigData/hadoop/data1/nm/containerlogs/
application_1449841777199_0003/container_1449841777199_0003_02_000001 -
Dyarn.app.container.log.filesize=0 -Dhadoop.root.logger=INFO,CLA
-Dhadoop.root.logfile=syslog -Xmx784m org.apache.hadoop.mapreduce.v2.app.MRAppMaster
|- 21557 21547 21557 21557 (bash) 0 0 13074432 368 /bin/bash -c ${BIGDATA_HOME}/jdk1.8.0_51//bin/
java
-Djava.io.tmpdir=/srv/BigData/hadoop/data1/nm/localdir/usercache/hbase/appcache/
application_1449841777199_0003/container_1449841777199_0003_02_000001/tmp -
Dlog4j.configuration=container-log4j.properties
-Dyarn.app.container.log.dir=/srv/BigData/hadoop/data1/nm/containerlogs/
application_1449841777199_0003/container_1449841777199_0003_02_000001 -
Dyarn.app.container.log.filesize=0 -Dhadoop.root.logger=INFO,CLA
-Dhadoop.root.logfile=syslog -Xmx784m org.apache.hadoop.mapreduce.v2.app.MRAppMaster 1>/srv/
BigData/hadoop/data1/nm/containerlogs/application_1449841777199_0003/
container_1449841777199_0003_02_000001/stdout
2>/srv/BigData/hadoop/data1/nm/containerlogs/application_1449841777199_0003/
container_1449841777199_0003_02_000001/stderr
Container killed on request. Exit code is 143
Container exited with a non-zero exit code 143
Failing this attempt. Failing the application.
```

Answer

This is a performance specification problem. The root cause of the MapReduce task execution failure is the memory overflow of ApplicationMaster, that is, the NodeManager kills the task due to the physical memory overflow.

Solutions:

Increase the memory of ApplicationMaster and optimize the following parameters in the *client installation path*/**Yarn/config/mapred-site.xml** configuration file on the client:

- **yarn.app.mapreduce.am.resource.mb**
- **yarn.app.mapreduce.am.command-opts**. The recommended value of **-Xmx** is 0.8 x **yarn.app.mapreduce.am.resource.mb**.

Specification:

ApplicationMaster supports 24,000 concurrent containers when the configuration is as follows:

- **yarn.app.mapreduce.am.resource.mb**=2048
- In **yarn.app.mapreduce.am.command-opts**, **-Xmx** is **1638m**.

21.8.7 After the Address of MapReduce JobHistoryServer Is Changed, Why the Wrong Page is Displayed When I Click the Tracking URL on the ResourceManager WebUI?

Question

After the address of MapReduce JobHistoryServer is changed, why the wrong page is displayed when I click the tracking URL on the ResourceManager WebUI?

Answer

JobHistoryServer address (mapreduce.jobhistory.address / mapreduce.jobhistory.webapp.<https.>address) is the parameter of MapReduce. The MapReduce client will submit the address together with jobs to ResourceManager. After ResourceManager completing the jobs, the parameter is saved in RMStateStore as the target address for viewing history job information.

If the JobHistoryServer address is changed, update the address in the configuration file of the MapReduce client in time. If the address is not updated, the page of earlier JobHistoryServer is displayed when you click the tracking URL of the new job. The target address of information about MapReduce jobs running before the change of address cannot be changed, so the wrong page is also displayed when you click the tracking URL. You can check the history information by accessing the new JobHistoryServer address.

21.8.8 MapReduce Job Failed in Multiple NameService Environment

Question

MapReduce or Yarn job fails in multiple nameService environment using viewFS.

Answer

When using viewFS only the mount directories are accessible, so the most possible cause is that the path configured is not in one of the mounted paths. For example:

```
<property>
<name>fs.defaultFS</name>
<value>viewfs://ClusterX/</value>
</property>
<property>
<name>fs.viewfs.mounttable.ClusterX.link./folder1</name>
<value>hdfs://NS1/folder1</value>
</property>
<property>
<name>fs.viewfs.mounttable.ClusterX.link./folder2</name>
<value>hdfs://NS2/folder2</value>
</property>
```


For all the MR properties which depends on HDFS, should use the paths inside mount folders.

Incorrect:

```
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>/tmp/hadoop-yarn/staging</value>
</property>
```

As the root folder (/) is not accessible in viewFS.

Correct:

```
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>/folder1/tmp/hadoop-yarn/staging</value>
</property>
```

21.8.9 Why a Fault MapReduce Node Is Not Blacklisted?

Question

MapReduce task fails and the ratio of fault nodes to all nodes is smaller than the blacklist threshold configured by **yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold**. Why the fault node not be blacklisted?

Answer

If the blacklisted percentage exceeds the threshold, all blacklisted nodes are released. Traditionally, the blacklist percentage is the ratio of fault nodes to all nodes in the cluster. Currently, each node has a label expression. Therefore, the blacklist percentage needs to be calculated based on the number of nodes related to valid node label expressions. In other way, the blacklist percentage is the ratio of fault nodes related to valid node label expressions.

Assume that there are 100 nodes in the cluster, including 10 nodes (labelA) related to valid node label expressions. Assume that all nodes related to valid node label expressions are faulty and default blacklist threshold is 0.33. In traditional calculation method, 10/100 = 0.1, which is far smaller than the threshold (0.33). In this case, the 10 nodes will never get released. Therefore, MapReduce always cannot obtain nodes and applications cannot run properly. In practice, the blacklist percentage needs to be calculated based on the total number of nodes related to valid node label expressions: 10/10 = 1 is greater than the blacklist threshold and all nodes are released.

Therefore, even the ratio of fault nodes to all nodes in the cluster is below the threshold, all nodes in the blacklist are released.

22 Using Oozie

22.1 Submitting a Job Using the Oozie Client

22.1.1 Oozie Client Configurations

Scenario

This section describes how to use the Oozie client in an O&M scenario or service scenario. Oozie can submit multiple types of tasks, such as Hive, Spark2x, Loader, MapReduce, Java, DistCp, Shell, HDFS, SSH, SubWorkflow, Streaming, and scheduled tasks.

Prerequisites

- The client has been installed in a directory, for example, **/opt/client**. For details, see **Installing a Client**. The client directory in the following operations is only an example. Change it based on site requirements.
- Service component users have been created by the MRS cluster administrator. In security mode, machine-machine users need to download the keytab file. A human-machine user must change the password upon the first login.

Using the Oozie Client

Step 1 Log in to the node where the client is installed as the client installation user.

Step 2 Run the following command to switch to the client installation directory (change it to the actual installation directory):

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Check the cluster authentication mode.

- If the cluster is in security mode, run the following command to authenticate the user: *exampleUser* indicates the name of the user who submits tasks.

kinit *exampleUser*

- If the cluster is in normal mode, go to **Step 5**.

Step 5 Perform the following operations to configure Hue:

1. Configure the Spark2x environment (skip this step if the Spark2x task is not involved):

   hdfs dfs -put /opt/client/Spark2x/spark/jars/*.jar /user/oozie/share/lib/spark2x/

   When the JAR package in the HDFS directory **/user/oozie/share** changes, you need to restart the Oozie service.

2. Upload the Oozie configuration file and JAR package to HDFS.

   hdfs dfs -mkdir /user/*exampleUser*

   hdfs dfs -put -f /opt/client/Oozie/oozie-client-*/examples /user/*exampleUser*/

   NOTE

   - *exampleUser* indicates the name of the user who submits tasks.
   - If the user who submits the task and other files except **job.properties** are not changed, client installation directory **Oozie/oozie-client-*/examples** can be repeatedly used after being uploaded to HDFS.
   - Resolve the JAR file conflict between Spark and Yarn about Jetty.

     hdfs dfs -rm -f /user/oozie/share/lib/spark/jetty-all-9.2.22.v20170606.jar
   - In normal mode, if **Permission denied** is displayed during the upload, run the following commands:

     su - omm

     source /opt/client/bigdata_env

     hdfs dfs -chmod -R 777 /user/oozie

     exit

Step 6 The following describes how to submit a MapReduce job on the Oozie client.

1. Run the following commands to modify the job execution configuration file:

   cd /opt/client/Oozie/oozie-client-*/examples/apps/map-reduce/

   vi job.properties

   nameNode=hdfs://hacluster
   resourceManager=*10.64.35.161*:8032 (**10.64.35.161** is the service plane IP address of the Yarn resourceManager (active) node, and 8032 is the port number of **yarn.resourcemanager.port**)
   queueName=default
   examplesRoot=examples
   user.name=admin
   oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/apps/map-reduce
   # HDFS upload path
   outputDir=map-reduce
   oozie.wf.rerun.failnodes=true

2. Run the following command to execute the Oozie job:

   oozie job -oozie https://*Host name of the Oozie role***:21003/oozie/ -config job.properties -run**

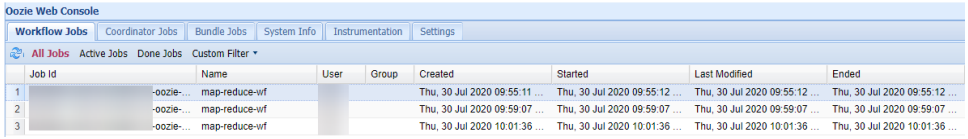
   21003 indicates the running port of Oozie HTTPS requests. You can log in to FusionInsight Manager, choose **Cluster** > **Services** > **Oozie** > **Configurations**, and search for **OOZIE_HTTPS_PORT** in the search box.

   [root@kwephispra44947 map-reduce]# oozie job -oozie https://kwephispra44948:21003/oozie/ -config job.properties -run


```
.....
job: 0000000-200730163829770-oozie-omm-W
```

3. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
4. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Oozie**, click the hyperlink next to **Oozie WebUI** to go to the Oozie page, and view the task execution result on the Oozie web UI.

Figure 22-1 Task execution result



----End

22.1.2 Submitting a Hive Task Using the Oozie Client

Scenario

This section describes how to use the Oozie client to submit a Hive job.

Hive jobs are divided into the following types:

- Hive job

  Hive job that is connected in JDBC mode
- Hive2 job

  Hive job that is connected in Beeline mode

This section describes how to submit a Hive job using the Oozie client.

NOTE

- The procedure for submitting a Hive2 job using the Oozie client is the same as that for submitting a Hive job. You only need to change **/Hive** in the procedure to **/Hive2**.

  For example, the directory of Hive jobs is **/opt/client/Oozie/oozie-client-*/examples/apps/hive/**, and that of Hive2 jobs is **/opt/client/Oozie/oozie-client-*/examples/apps/hive2/**.
- You are advised to download the latest client.

Prerequisites

- The Hive and Oozie components and clients have been installed and are running properly.
- You have created or obtained the human-machine account and password for accessing the Oozie service.

NOTE

- This user must belong to the **hadoop**, **supergroup**, and **hive** groups and be assigned with the Oozie role operation permission. If the multi-instance function is enabled for Hive, the user must belong to a specific Hive instance group, for example, **hive3**.
- This user must also be assigned the **manager_viewer** role at least.

- You have obtained the URL of the Oozie server (any instance) in the running state, for example, **https://10.1.130.10:21003/oozie**.
- You have obtained the name of the Oozie server, for example, **10-1-130-10**.
- You have obtained the IP address of the active Yarn ResourceManager, for example, **10.1.130.11**.

Procedure

Step 1 Log in to the node where the Oozie client is installed as the client installation user.

Step 2 Run the following command to obtain the installation environment. **/opt/client** is an example client installation path.

source /opt/client/bigdata_env

Step 3 Check the cluster authentication mode.

- If the cluster is in security mode, run the **kinit** command to authenticate users.

  For example, the **oozieuser** user is authenticated using the following command:

  kinit oozieuser

- If the cluster is in normal mode, go to **Step 4**.

Step 4 Run the following command to go to the example directory:

cd /opt/client/Oozie/oozie-client-*/examples/apps/hive/

Table 22-1 lists the files that you need to pay attention to in the directory.

Table 22-1 File description

File	Description
hive-site.xml	Configuration file of a Hive job
job.properties	Parameter definition file of a workflow
script.q	SQL script of a Hive job
workflow.xml	Rule definition file of a workflow

Step 5 Run the following command to edit the **job.properties** file:

vi job.properties

Perform the following modifications:

Change the value of **userName** to the name of the human-machine user who submits the job, for example, **userName=oozieuser**.

Step 6 Run the **oozie job** command to run the workflow file:

oozie job -oozie https://*Host name of the Oozie role*:**21003/oozie/ -config job.properties -run**

NOTE

- The command parameters are described as follows:

  -oozie URL of the Oozie server that executes a job

  -config Workflow property file

  -run Executing a workflow

- If a job ID, for example, **job: 0000021-140222101051722-oozie-omm-W**, is displayed after the workflow file is executed, the job is successfully submitted. You can view the execution results on the Oozie management page.

  Log in to the Oozie web UI at **https**://*IP address of the Oozie role*:**21003/oozie** as user **oozieuser**.

  On the Oozie web UI, you can view the submitted workflow information based on the job ID in the table on the page.

----End

22.1.3 Submitting a Spark2x Task Using the Oozie Client

Scenario

This section describes how to submit a Spark2x job using the Oozie client.

NOTE

You are advised to download the latest client.

Prerequisites

- The Spark2x and Oozie components and clients have been installed and are running properly.

  If the current client is an earlier version, you need to download and install the client again.

- You have created or obtained the human-machine account and password for accessing the Oozie service.

  NOTE

  - This user must belong to the **hadoop**, **supergroup**, and **hive** groups and be assigned with the Oozie role operation permission. If the multi-instance function is enabled for Hive, the user must belong to a specific Hive instance group, for example, **hive3**.
  - This user must also be assigned the **manager_viewer** role at least.

- You have obtained the URL of the Oozie server (any instance) in the running state, for example, **https://10.1.130.10:21003/oozie**.
- You have obtained the name of the Oozie server, for example, **10-1-130-10**.

- You have obtained the IP address of the active Yarn ResourceManager, for example, **10.1.130.11**.

Procedure

Step 1 Log in to the node where the Oozie client is installed as the client installation user.

Step 2 Run the following command to obtain the installation environment. **/opt/client** is an example client installation path.

source /opt/client/bigdata_env

Step 3 Check the cluster authentication mode.

- If the cluster is in security mode, run the **kinit** command to authenticate users.

  For example, the **oozieuser** user is authenticated using the following command:

  kinit oozieuser

- If the cluster is in normal mode, go to **Step 4**.

Step 4 Run the following command to go to the example directory:

cd /opt/client/Oozie/oozie-client-*/examples/apps/spark2x/

Table 22-2 lists the files that you need to pay attention to in the directory.

Table 22-2 File description

File	Description
job.properties	Parameter definition file of a workflow
workflow.xml	Rule definition file of a workflow
lib	Directory of the JAR file on which a workflow depends

Step 5 Run the following command to edit the **job.properties** file:

vi job.properties

Perform the following modifications:

Change the value of **userName** to the name of the human-machine user who submits the job, for example, **userName=oozieuser**.

Step 6 Run the **oozie job** command to run the workflow file:

oozie job -oozie https://*Host name of the Oozie role***:21003/oozie/ -config job.properties -run**

NOTE

- The command parameters are described as follows:

  -oozie URL of the Oozie server that executes a job

  -config Workflow property file

  -run Executing a workflow

- If a job ID, for example, **job: 0000021-140222101051722-oozie-omm-W**, is displayed after the workflow file is executed, the job is successfully submitted. You can view the execution results on the Oozie management page.

  Log in to the Oozie web UI at **https**://*IP address of the Oozie role*:**21003/oozie** as user **oozieuser**.

  On the Oozie web UI, you can view the submitted workflow information based on the job ID in the table on the page.

----End

22.1.4 Submitting a Loader Task Using the Oozie Client

Scenario

This section describes how to submit a Loader job using the Oozie client.

NOTE

You are advised to download the latest client.

Prerequisites

- The Hive and Oozie components and clients have been installed and are running properly.
- You have created or obtained the human-machine account and password for accessing the Oozie service.

  NOTE

  - This user must belong to the **hadoop**, **supergroup**, and **hive** groups and be assigned with the Oozie role operation permission. If the multi-instance function is enabled for Hive, the user must belong to a specific Hive instance group, for example, **hive3**.
  - This user must also be assigned the **manager_viewer** role at least.
- You have obtained the URL of the Oozie server (any instance) in the running state, for example, **https://10.1.130.10:21003/oozie**.
- You have obtained the name of the Oozie server, for example, **10-1-130-10**.
- You have obtained the IP address of the active Yarn ResourceManager, for example, **10.1.130.11**.
- You have created a Loader job to be scheduled and obtained the job ID.

Procedure

Step 1 Log in to the node where the Oozie client is installed as the client installation user.

Step 2 Run the following command to obtain the installation environment. **/opt/client** is an example client installation path.

source /opt/client/bigdata_env

Step 3 Check the cluster authentication mode.

- If the cluster is in security mode, run the **kinit** command to authenticate users.

  For example, the **oozieuser** user is authenticated using the following command:

  kinit oozieuser

- If the cluster is in normal mode, go to **Step 4**.

Step 4 Run the following command to go to the example directory:

cd /opt/client/Oozie/oozie-client-*/examples/apps/sqoop/

Table 22-3 lists the files that you need to pay attention to in the directory.

Table 22-3 File description

File	Description
job.properties	Parameter definition file of a workflow
workflow.xml	Rule definition file of a workflow

Step 5 Run the following command to edit the **job.properties** file:

vi job.properties

Perform the following modifications:

Change the value of **userName** to the name of the human-machine user who submits the job, for example, **userName=oozieuser**.

Step 6 Run the following command to edit the **workflow.xml** file:

vi workflow.xml

Perform the following modifications:

Change the value of **command** to the ID of the Loader job to be scheduled, for example, **1**.

Upload the **workflow.xml** file to the HDFS path in the **job.properties** file.

hdfs dfs -put -f workflow.xml /user/userName/examples/apps/sqoop

Step 7 Run the **oozie job** command to run the workflow file:

oozie job -oozie https://*Host name of the Oozie role***:21003/oozie/ -config job.properties -run**

NOTE

- The command parameters are described as follows:

  -oozie URL of the Oozie server that executes a job

  -config Workflow property file

  -run Executing a workflow

- If a job ID, for example, **job: 0000021-140222101051722-oozie-omm-W**, is displayed after the workflow file is executed, the job is successfully submitted. You can view the execution results on the Oozie management page.

  Log in to the Oozie web UI at **https**:///*IP address of the Oozie role*:**21003/oozie** as user **oozieuser**.

  On the Oozie web UI, you can view the submitted workflow information based on the job ID in the table on the page.

----End

22.1.5 Submitting a DistCp Task Using the Oozie Client

Scenario

This section describes how to submit a DistCp job using the Oozie client.

NOTE

You are advised to download the latest client.

Prerequisites

- The HDFS and Oozie components and clients have been installed and are running properly.

  If the current client is an earlier version, you need to download and install the client again.

- You have created or obtained the human-machine account and password for accessing the Oozie service.

  NOTE

  - This user must belong to the **hadoop**, **supergroup**, and **hive** groups and be assigned with the Oozie role operation permission. If the multi-instance function is enabled for Hive, the user must belong to a specific Hive instance group, for example, **hive3**.
  - This user must also be assigned the **manager_viewer** role at least.

- You have obtained the URL of the Oozie server (any instance) in the running state, for example, **https://10.1.130.10:21003/oozie**.
- You have obtained the name of the Oozie server, for example, **10-1-130-10**.
- You have obtained the IP address of the active Yarn ResourceManager, for example, **10.1.130.11**.

Procedure

Step 1 Log in to the node where the Oozie client is installed as the client installation user .

Step 2 Run the following command to obtain the installation environment. **/opt/client** is an example client installation path.

source /opt/client/bigdata_env

Step 3 Check the cluster authentication mode.

- If the cluster is in security mode, run the **kinit** command to authenticate users.

  For example, the **oozieuser** user is authenticated using the following command:

  kinit oozieuser

- If the cluster is in normal mode, go to **Step 4**.

Step 4 Run the following command to go to the example directory:

cd /opt/client/Oozie/oozie-client-*/examples/apps/distcp/

Table 22-4 lists the files that you need to pay attention to in the directory.

Table 22-4 File description

File	Description
job.properties	Parameter definition file of a workflow
workflow.xml	Rule definition file of a workflow

Step 5 Run the following command to edit the **job.properties** file:

vi job.properties

Perform the following modifications:

Change the value of **userName** to the name of the human-machine user who submits the job, for example, **userName=oozieuser**.

Step 6 Whether DistCp is not deployed across security clusters.

- If yes, go to **Step 7**.
- If no, go to **Step 9**.

Step 7 Establish cross-Manager mutual trust between two clusters.

Step 8 Run the following commands to back up and modify the **workflow.xml** file:

cp workflow.xml workflow.xml.bak

vi workflow.xml

Modify the following content:

```
<workflow-app xmlns="uri:oozie:workflow:1.0" name="distcp-wf">
   <start to="distcp-node"/>
   <action name="distcp-node">
      <distcp xmlns="uri:oozie:distcp-action:1.0">
         <resource-manager>${resourceManager}</resource-manager>
         <name-node>${nameNode}</name-node>
         <prepare>
            <delete path="hdfs://target_ip:target_port/user/${userName}/${examplesRoot}/output-data/$
```



```
{outputDir}"/>
            </prepare>
            <configuration>
               <property>
                  <name>mapred.job.queue.name</name>
                  <value>${queueName}</value>
               </property>
               <property>
                  <name>oozie.launcher.mapreduce.job.hdfs-servers</name>
                  <value>hdfs://source_ip:source_port,hdfs://target_ip:target_port</value>
               </property>
            </configuration>
            <arg>${nameNode}/user/${userName}/${examplesRoot}/input-data/text/data.txt</arg>
            <arg>hdfs://target_ip:target_port/user/${userName}/${examplesRoot}/output-data/${outputDir}/
data.txt</arg>
         </distcp>
      <ok to="end"/>
      <error to="fail"/>
   </action>
   <kill name="fail">
      <message>DistCP failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
   </kill>
   <end name="end"/>
</workflow-app>
```

target_ip:target_port is the HDFS active NameNode address of the other trusted cluster, for example, **10.10.10.233:25000**.

source_ip:source_port indicates the HDFS active NameNode address of the source cluster, for example, **10.10.10.223:25000**.

Change the two IP addresses and port numbers based on the site requirements.

Step 9 Run the **oozie job** command to run the workflow file:

oozie job -oozie https://*Host name of the Oozie role***:21003/oozie/ -config job.properties -run**

NOTE

- The command parameters are described as follows:

  -oozie URL of the Oozie server that executes a job

  -config Workflow property file

  -run Executing a workflow

- If a job ID, for example, **job: 0000021-140222101051722-oozie-omm-W**, is displayed after the workflow file is executed, the job is successfully submitted. You can view the execution results on the Oozie management page.

  Log in to the Oozie web UI at **https**://*IP address of the Oozie role***:21003/oozie** as user **oozieuser**.

  On the Oozie web UI, you can view the submitted workflow information based on the job ID in the table on the page.

----End

22.1.6 Submitting Other Tasks Using the Oozie Client

Scenario

In addition to Hive, Spark2x, and Loader jobs, MapReduce, Java, Shell, HDFS, SSH, SubWorkflow, Streaming, and scheduled jobs can be submitted using the Oozie client.

NOTE

You are advised to download the latest client.

Prerequisites

- The Oozie component and its client have been installed and are running properly.
- You have created or obtained the human-machine account and password for accessing the Oozie service.

  NOTE

  - Shell job:

    This user must belong to the **hadoop** and **supergroup** groups and be assigned the Oozie role operation permission. The Shell script must have the execution permission on each NodeManager.
  - SSH job:

    This user must belong to the **hadoop** and **supergroup** groups and be assigned the Oozie role operation permission. The mutual trust configuration is complete.
  - Other jobs:

    This user must belong to the **hadoop** and **supergroup** groups and be assigned the Oozie role operation permission and other required permissions.
  - This user must also be assigned the **manager_viewer** role at least.
- You have obtained the URL of the Oozie server (any instance) in the running state, for example, **https://10.1.130.10:21003/oozie**.
- You have obtained the name of the Oozie server, for example, **10-1-130-10**.
- You have obtained the IP address of the active Yarn ResourceManager, for example, **10.1.130.11**.

Procedure

Step 1 Log in to the node where the Oozie client is installed as the client installation user.

Step 2 Run the following command to obtain the installation environment. **/opt/client** is an example client installation path.

source /opt/client/bigdata_env

Step 3 Check the cluster authentication mode.

- If the cluster is in security mode, run the **kinit** command to authenticate users.

  For example, the **oozieuser** user is authenticated using the following command:

  kinit oozieuser
- If the cluster is in normal mode, go to **Step 4**.

Step 4 Go to the example directory based on the type of the task you submit.

Table 22-5 List of example directories

Job Type	Example Directory
MapReduce job	*Client installation directory*/**Oozie/oozie-client-*/examples/apps/map-reduce**
Java job	*Client installation directory*/**Oozie/oozie-client-*/examples/apps/java-main**
Shell job	*Client installation directory*/**Oozie/oozie-client-*/examples/apps/shell**
Streaming job	*Client installation directory*/**Oozie/oozie-client-*/examples/apps/shell**
SubWorkflow job	*Client installation directory*/**Oozie/oozie-client-*/examples/apps/subwf**
SSH job	*Client installation directory*/**Oozie/oozie-client-*/examples/apps/ssh**
Scheduled job	*Client installation directory*/**Oozie/oozie-client-*/examples/apps/cron**

NOTE

The examples of other jobs contain HDFS job examples.

Table 22-6 lists the files that you need to pay attention to in the example directory.

Table 22-6 File description

File	Description
job.properties	Parameter definition file of a workflow
workflow.xml	Rule definition file of a workflow
lib	Directory of the JAR file on which a workflow depends
coordinator.xml	Scheduled job configuration file which can be used to set a scheduled policy. The file is in the **cron** directory.
oozie_shell.sh	Shell script file required for submitting shell jobs. The file is in the **shell** directory.

Step 5 Run the following command to edit the **job.properties** file:

vi job.properties

Perform the following modifications:

Change the value of **userName** to the name of the human-machine user who submits the job, for example, **userName=oozieuser**.

Step 6 Run the **oozie job** command to run the workflow file:

oozie job -oozie https://*Host name of the oozie role*:**21003/oozie -config** *File path of job.properties* **-run**

Example:

oozie job -oozie https://10-1-130-10:21003/oozie -config

/opt/client/Oozie/oozie-client-*/examples/apps/map-reduce/job.properties -run

NOTE

- The command parameters are described as follows:

  -oozie URL of the Oozie server that executes a job

  -config Workflow property file

  -run Executing a workflow
- If a job ID, for example, **job: 0000021-140222101051722-oozie-omm-W**, is displayed after the workflow file is executed, the job is successfully submitted. You can view the execution results on the Oozie management page.

  Log in to the Oozie web UI at **https**://*IP address of the Oozie role*:**21003/oozie** as user **oozieuser**.

  On the Oozie web UI, you can view the submitted workflow information based on the job ID in the table on the page.

----End

22.2 Using Hue to Submit an Oozie Job

22.2.1 Creating a Workflow Using Hue

Scenario

You can submit an Oozie job on the Hue management page, but a workflow must be created before the job is submitted.

Prerequisites

Before using Hue to submit an Oozie job, configure the Oozie client and upload the sample configuration file and JAR file to the specified HDFS directory. For details, see **Oozie Client Configurations**.

Procedure

Step 1 Prepare a user who has operation permissions on the corresponding components.

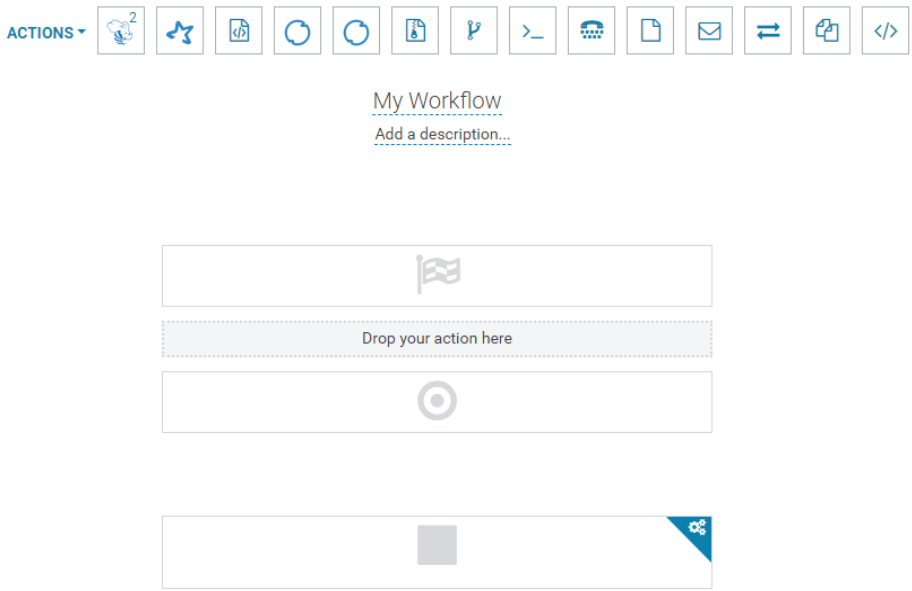
For example, log in to FusionInsight Manager as user **admin** and choose **System** in the top menu bar. On the **System** page that is displayed, choose **User** under **Permission** in the navigation pane on the left. On the displayed **User** page, click

Create. On the **Create** page, set **Username** to **hueuser** and **User Type** to **Human-Machine**, set the password and confirm it, set **User Group** to **hive**, **hadoop**, and **supergroup**, set **Primary Group** to **hive**, set **Role** to **System_administrator**, and click **OK**.

Step 2 Log in to FusionInsight Manager as the user created in **Step 1** (change the password upon your first login), choose **Cluster** > **Services** > **Hue**, and click the link next to **Hue WebUI** to go to the Hue WebUI page.

Step 3 In the navigation tree on the left, click  and choose **Workflow** to open the Workflow editor.

Step 4 Select **Actions** from the **DOCUMENTS** drop-down list, select the job type to be created and drag it to the operation area.



For submitting different job types, follow instructions in the following sections:

- **Submitting an Oozie Hive2 Job Using Hue**
- **Submitting an Oozie Spark2x Job Using Hue**
- **Submitting an Oozie Java Job Using Hue**
- **Submitting an Oozie Loader Job Using Hue**
- **Submitting an Oozie MapReduce Job Using Hue**
- **Submitting an Oozie Sub-workflow Job Using Hue**
- **Submitting an Oozie Shell Job Using Hue**
- **Submitting an Oozie HDFS Job Using Hue**
- **Submitting an Oozie Streaming Job Using Hue**
- **Submitting an Oozie DistCp Job Using Hue**

----End

22.2.2 Submitting an Oozie Hive2 Job Using Hue

Scenario

This section describes how to submit an Oozie job of the Hive2 type on the Hue web UI.

Procedure

Step 1 Create a workflow. For details, see **Creating a Workflow Using Hue**.

Step 2 On the workflow editing page, select  next to **HiveServer2 Script** and drag it to the operation area.

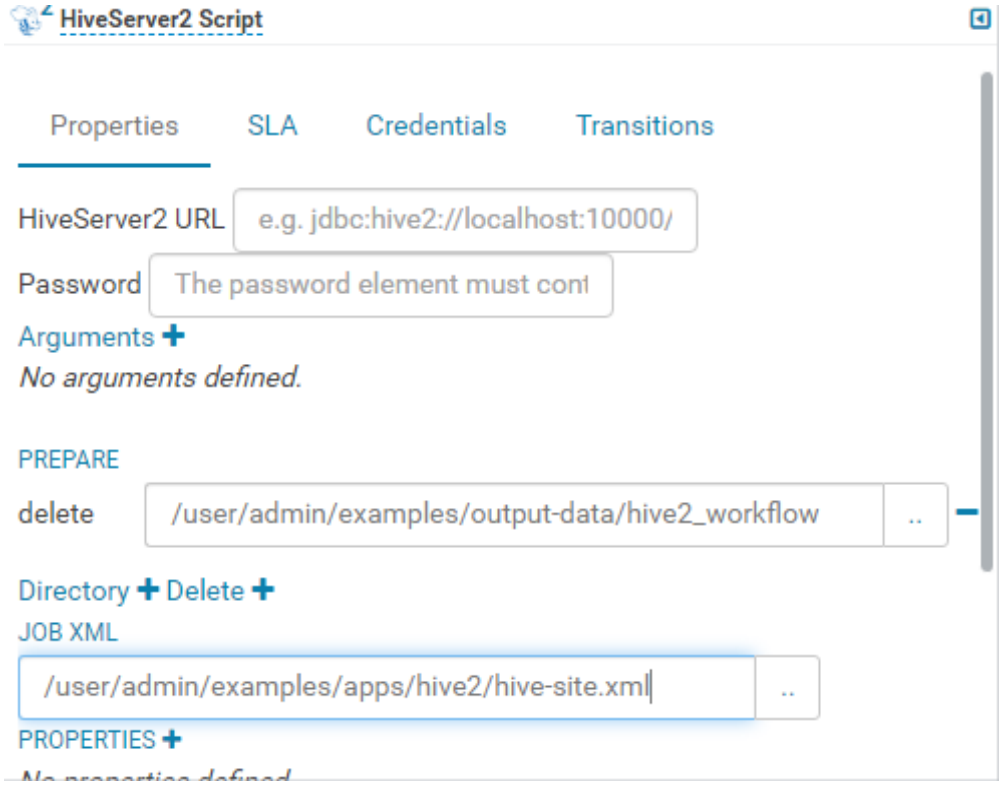
Step 3 In the **HiveServer2 Script** dialog box that is displayed, configure the script path in the HDFS, for example, **/user/admin/examples/apps/hive2/script.q**, and click **Add**.

Step 4 Click **PARAMETER+** to add input and output parameters.

For example, if the input parameter is **INPUT=/user/admin/examples/input-data/table**, the output parameter is **OUTPUT=/user/admin/examples/output-data/hive2_workflow**.

Step 5 Click the configuration button  in the upper right corner. On the configuration page that is displayed, click **Delete +** to delete a directory, for example, **/user/admin/examples/output-data/hive2_workflow**.

Step 6 Configure **JOB XML**. The value is *the path where the client installation directory*/**Oozie/oozie-client-*/examples/apps/hive/hive-site.xml** is uploaded to the HDFS directory, for example, to the HDFS path **/user/admin/examples/apps/hive2/hive-site.xml**. You do not need to set **HiveServer2 URL** and other parameters.



NOTE

If the preceding parameters and values are modified, you can query them in **Oozie client installation directory//oozie-client-*/conf/hive-site.xml**.

Step 7 Click in the upper right corner of the Oozie editor.

If you need to modify the job name before saving the job (default value: **My Workflow**), click the name directly for modification, for example, **Hive2-Workflow**.

Step 8 After the configuration is saved, click , and submit the job.

After the job is submitted, you can view the related contents of the job, such as the detailed information, logs, and processes, on Hue.

----End

22.2.3 Submitting an Oozie HQL Script Using Hue

Scenario

This section describes how to submit a Hive job on the Hue web UI.

Procedure

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 In the navigation tree on the left, click and choose **Workflow** to open the Workflow editor.

Step 3 Click **Documents**, click  to select a Hive script from the operation list, and drag it to the operation page.

Step 4 In the **HiveServer2 Script** dialog box that is displayed, select the saved Hive script. For details about how to save the Hive script, see **Using Hue to Execute HiveQL**. Select a script and click **Add**.

Step 5 Configure the Job XML, for example, to the HDFS path **/user/admin/examples/apps/hive2/hive-site.xml**. For details, see **Submitting an Oozie Hive2 Job Using Hue**.

Step 6 Click  in the upper right corner of the Oozie editor.

Step 7 After the configuration is saved, click  , and submit the job.

After the job is submitted, you can view the related contents of the job, such as the detailed information, logs, and processes, on Hue.

----End

22.2.4 Submitting an Oozie Spark2x Job Using Hue

Scenario

This section describes how to submit an Oozie job of the Spark2x type on Hue.

Procedure

Step 1 Create a workflow. For details, see **Creating a Workflow Using Hue**.

Step 2 On the workflow editing page, select  next to **Spark program** and drag it to the operation area.

Step 3 In the Spark window that is displayed, set the value of **Files**, for example, to **hdfs://hacluster/user/admin/examples/apps/spark2x/lib/oozie-examples.jar**. Set the value of **jar/py name**, for example, to **org.apache.oozie.example.SparkFileCopy**, and click **Add**.

Step 4 Set the value of **Main class**, for example, **org.apache.oozie.example.SparkFileCopy**.

Step 5 Click **PARAMETER+** to add related input and output parameters.

For example, add the following parameters:

- **hdfs://hacluster/user/admin/examples/input-data/text/data.txt**
- **hdfs://hacluster/user/admin/examples/output-data/spark_workflow**

Step 6 In the **Options list** text box, specify Spark parameters, for example, **--conf spark.yarn.archive=hdfs://hacluster/user/spark2x/jars/*xxx*/spark-archive-2x.zip --conf spark.eventLog.enabled=true --conf spark.eventLog.dir=hdfs://hacluster/spark2xJobHistory2x**.

NOTE

The version *xxx* is used as an example. You can log in to FusionInsight Manager, click  in the upper right corner, choose **About** from the drop-down list, and view the FusionInsight Manager version in the dialog box that is displayed.

Step 7 Click the configuration button  in the upper right corner. Set the value of **Spark Master**, for example, to **yarn-cluster**. Set the value of **Mode**, for example, **cluster**.

Step 8 On the configuration page that is displayed, click **Delete +** to delete a directory, for example, **hdfs://hacluster/user/admin/examples/output-data/spark_workflow**.

Step 9 Click **PROPERTIES+** and add **sharelib** used by Oozie. Enter the attribute name **oozie.action.sharelib.for.spark** in the left text box and the attribute value **spark2x** in the right text box.

Step 10 Click  in the upper right corner of the Oozie editor.

If you need to modify the job name before saving the job (default value: **My Workflow**), click the name directly for modification, for example, **Spark-Workflow**.

Step 11 After the configuration is saved, click  , and submit the job.

After the job is submitted, you can view the related contents of the job, such as the detailed information, logs, and processes, on Hue.

----End

22.2.5 Submitting an Oozie Java Job Using Hue

Scenario

This section describes how to submit an Oozie job of the Java type on the Hue web UI.

Procedure

Step 1 Create a workflow. For details, see **Creating a Workflow Using Hue**.

Step 2 On the workflow editing page, select  next to **Java program** and drag it to the operation area.

Step 3 In the **Jar program** window that is displayed, set the value of **Jar name**, for example, **/user/admin/examples/apps/java-main/lib/oozie-examples-5.1.0.jar**. Set the value of **Main class**, for example, **org.apache.oozie.example.DemoJavaMain**. Click **Add**.

Step 4 Click  in the upper right corner of the Oozie editor.

If you need to modify the job name before saving the job (default value: **My Workflow**), click the name directly for modification, for example, **Java-Workflow**.

Step 5 After the configuration is saved, click  , and submit the job.

After the job is submitted, you can view the related contents of the job, such as the detailed information, logs, and processes, on Hue.

----End

22.2.6 Submitting an Oozie Loader Job Using Hue

Scenario

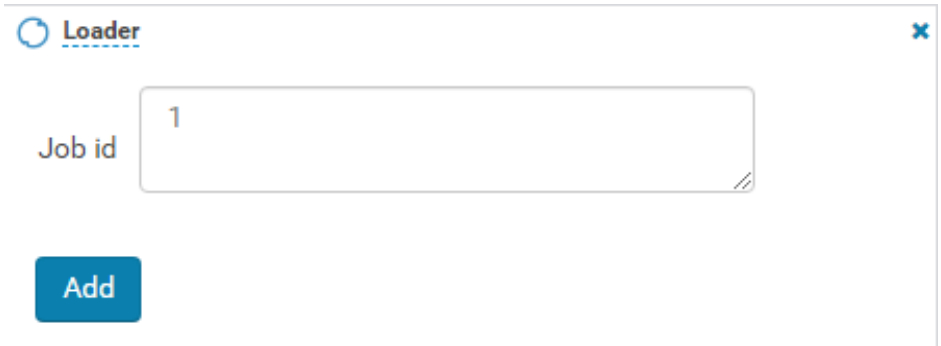
This section describes how to submit an Oozie job of the Loader type on the Hue web UI.

Procedure

Step 1 Create a workflow. For details, see **Creating a Workflow Using Hue**.

Step 2 On the workflow editing page, select  next to **Loader** and drag it to the operation area.

Step 3 In the **Loader** window that is displayed, set **Job id**, for example, to **1**. Click **Add**.



NOTE

Job id is the ID of the Loader job to be orchestrated and can be obtained from the Loader page.

You can create a Loader job to be scheduled and obtain its job ID. For details, see **Using Loader**.

Step 4 Click in the upper right corner of the Oozie editor.

If you need to modify the job name before saving the job (default value: **My Workflow**), click the name directly for modification, for example, **Loader-Workflow**.

Step 5 After the configuration is saved, click , and submit the job.

After the job is submitted, you can view the related contents of the job, such as the detailed information, logs, and processes, on Hue.

----End

22.2.7 Submitting an Oozie MapReduce Job Using Hue

Scenario

This section describes how to submit an Oozie job of the MapReduce type on the Hue web UI.

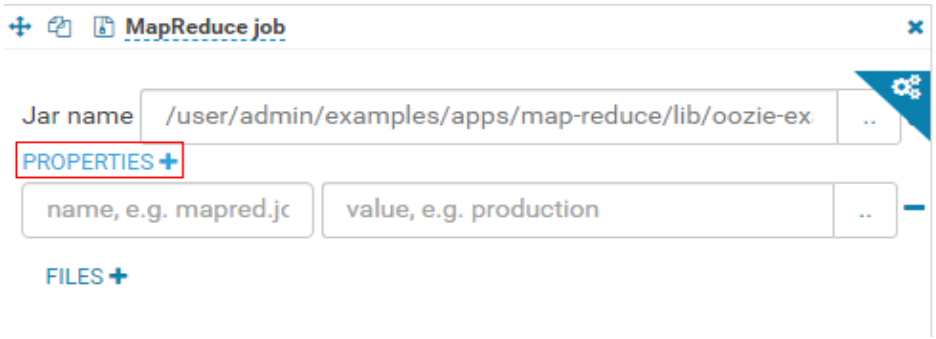
Procedure

Step 1 Create a workflow. For details, see **Creating a Workflow Using Hue**.

Step 2 On the workflow editing page, select next to **MapReduce job** and drag it to the operation area.

Step 3 In the displayed **MapReduce job** dialog box, set **Jar name**, for example, to **/user/admin/examples/apps/map-reduce/lib/oozie-examples-5.1.0.jar**. Click **Add**.

Step 4 Click **PROPERTIES+** to add input and output properties.



For example, set the value of **mapred.input.dir** to **/user/admin/examples/input-data/text** and set the value of **mapred.output.dir** to **/user/admin/examples/output-data/map-reduce_workflow**.

Step 5 Click the configuration button  in the upper right corner. On the configuration page that is displayed, click **Delete +** to delete a directory, for example, **/user/admin/examples/output-data/map-reduce_workflow**.

Step 6 Click  in the upper right corner of the Oozie editor.

If you need to modify the job name before saving the job (default value: **My Workflow**), click the name directly for modification, for example, **MapReduce-Workflow**.

Step 7 After the configuration is saved, click  , and submit the job.

After the job is submitted, you can view the related contents of the job, such as the detailed information, logs, and processes, on Hue.

----End

22.2.8 Submitting an Oozie Sub-workflow Job Using Hue

Scenario

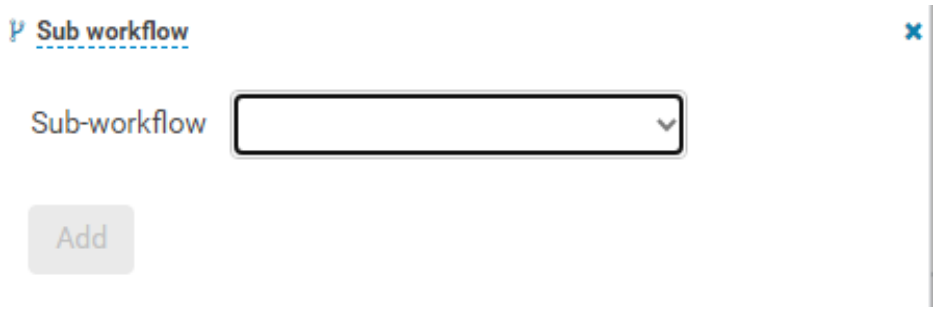
This section describes how to submit an Oozie job of the Sub-workflow type on the Hue web UI.

Procedure

Step 1 Create a workflow. For details, see **Creating a Workflow Using Hue**.

Step 2 On the workflow editing page, select  next to **Sub workflow** and drag it to the operation area.

Step 3 In the **Sub workflow** dialog box that is displayed, set **Sub-workflow**, for example, to **Java-Workflow** (one of the created workflows) from the drop-down list box, and click **Add**.



Step 4 Click in the upper right corner of the Oozie editor.

If you need to modify the job name before saving the job (default value: **My Workflow**), click the name directly for modification, for example, **Subworkflow-Workflow**.

Step 5 After the configuration is saved, click  , and submit the job.

After the job is submitted, you can view the related contents of the job, such as the detailed information, logs, and processes, on Hue.

----End

22.2.9 Submitting an Oozie Shell Job Using Hue

Scenario

This section describes how to submit an Oozie job of the Shell type on the Hue web UI.

Procedure

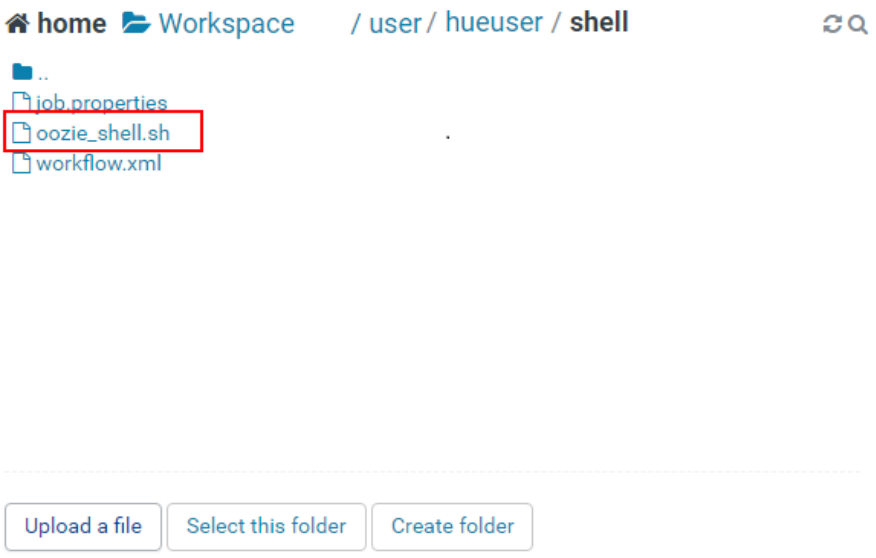
Step 1 Create a workflow. For details, see **Creating a Workflow Using Hue**.

Step 2 On the workflow editing page, select  next to **Shell** and drag it to the operation area.

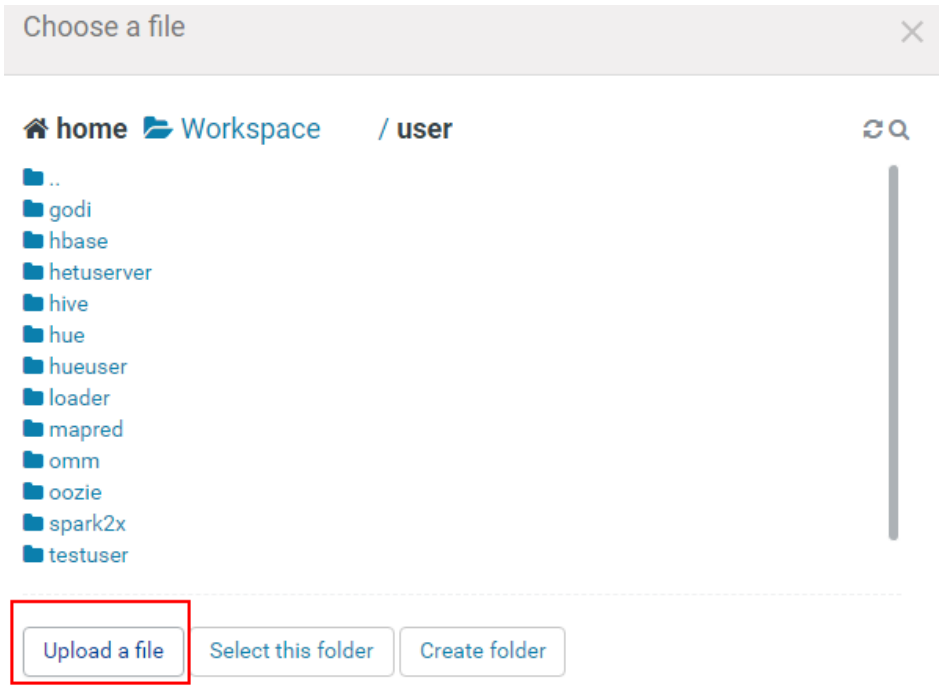
Step 3 In the **Shell** window that is displayed, set **Shell command**, for example, to **oozie_shell.sh**, and click **Add**.

Step 4 Click **FILE+** to add the Shell command execution file or Oozie example execution file. You can select a file stored in HDFS or a local file.

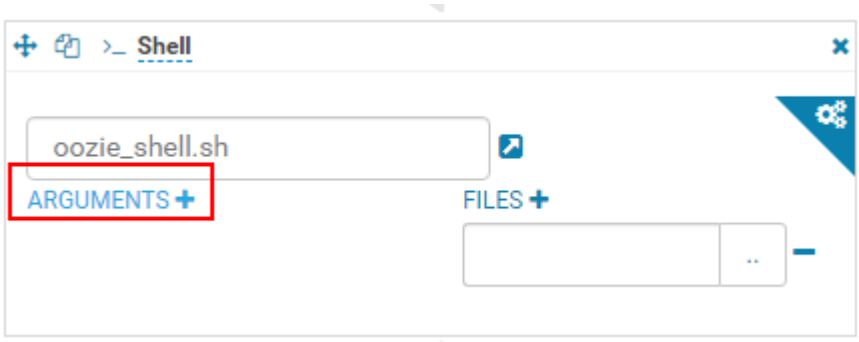
- If the file is stored in HDFS, select the path of the **.sh** file, for example, **user/hueuser/shell/oozie_shell.sh**.



- If you select a local file, click **Upload a file** on the **Choose a file** page to upload the local file. After the file is uploaded, select the file.



Step 5 If the shell file to be executed needs to transfer parameters, click **ARGUMENTS+** to set parameters.



NOTE

The sequence of transferring parameters must be the same as that in the shell script.

Step 6 Click in the upper right corner of the Oozie editor.

If you need to modify the job name before saving the job (default value: **My Workflow**), click the name directly for modification, for example, **Shell-Workflow**.

Step 7 After the configuration is saved, click , and submit the job.

After the job is submitted, you can view the related contents of the job, such as the detailed information, logs, and processes, on Hue.

NOTE

- When configuring a shell command as a Linux command, specify it as the original command instead of the shortcut key command. For example, do not set **ls -l** to **ll**. You can configure it as the shell command **ls**, and add a parameter **-l**.
- When uploading the shell script to HDFS on Windows, make sure that the shell script format is Unix. If the format is incorrect, the shell job fails to be submitted.

----End

22.2.10 Submitting an Oozie HDFS Job Using Hue

Scenario

This section describes how to submit an Oozie job of the HDFS type on the Hue web UI.

Procedure

Step 1 Create a workflow. For details, see **Creating a Workflow Using Hue**.

Step 2 On the workflow editing page, select  next to **Fs** and drag it to the operation area.

Step 3 In the **Fs** window that is displayed, click **Add**.

Step 4 Click **CREATE DIRECTORY+** to add the HDFS directories to be created, for example, **/user/admin/examples/output-data/mkdir_workflow** and **/user/admin/examples/output-data/mkdir_workflow1**.

CAUTION

If you click **DELETE PATH+** to add the HDFS path to be deleted, this parameter cannot be empty. Otherwise, the **/user**/*{Submit user name}* directory of the HDFS is deleted by default, which may cause other tasks to run abnormally.

Step 5 Click  in the upper right corner of the Oozie editor.

If you need to modify the job name before saving the job (default value: **My Workflow**), click the name directly for modification, for example, **HDFS-Workflow**.

Step 6 After the configuration is saved, click  , and submit the job.

After the job is submitted, you can view the related contents of the job, such as the detailed information, logs, and processes, on Hue.

----End

22.2.11 Submitting an Oozie Streaming Job Using Hue

Scenario

This section describes how to submit an Oozie job of the Streaming type on the Hue web UI.

Procedure

Step 1 Create a workflow. For details, see **Creating a Workflow Using Hue**.

Step 2 On the workflow editing page, select  next to **Streaming** and drag it to the operation area.

Step 3 In the **Streaming** window that is displayed, set **Mapper**, for example, to **/bin/cat**. Set **Reducer**, for example, to **/usr/bin/wc**. Click **Add**.

Step 4 Click **FILE+** to add the files required for running, for example, **/user/oozie/share/lib/mapreduce-streaming/hadoop-streaming-*xxx*.jar** and **/user/oozie/share/lib/mapreduce-streaming/oozie-sharelib-streaming-5.1.0.jar**.

Step 5 Click the configuration button  in the upper right corner. On the configuration page that is displayed, click **Delete+** to delete a directory, for example, **/user/admin/examples/output-data/streaming_workflow**.

Step 6 Click **PROPERTIES+** to add the following properties:

- Enter the property name **mapred.input.dir** in the left box and enter the property value **/user/admin/examples/input-data/text** in the right box.
- Enter the property name **mapred.output.dir** in the left box and enter the attribute value **/user/admin/examples/output-data/streaming_workflow** in the right box.

Step 7 Click  in the upper right corner of the Oozie editor.

If you need to modify the job name before saving the job (default value: **My Workflow**), click the name directly for modification, for example, **Streaming-Workflow**.

Step 8 After the configuration is saved, click  , and submit the job.

After the job is submitted, you can view the related contents of the job, such as the detailed information, logs, and processes, on Hue.

----End

22.2.12 Submitting an Oozie DistCp Job Using Hue

Scenario

This section describes how to submit an Oozie job of the DistCp type on the Hue web UI.

Procedure

Step 1 Create a workflow. For details, see **Creating a Workflow Using Hue**.

Step 2 On the workflow editing page, select  next to **Distcp** and drag it to the operation area.

Step 3 Determine whether the current DistCp operation is performed across clusters.

- If yes, go to **Step 4**.
- If no, go to **Step 7**.

Step 4 Establish cross-Manager mutual trust between two clusters.

Step 5 In the **Distcp** window that is displayed, set the value of **Source**, for example, to **hdfs://hacluster/user/admin/examples/input-data/text/data.txt**. Set **Destination**, for example, to **hdfs://target_ip:target_port/user/admin/examples/output-data/distcp-workflow/data.txt**. Click **Add**.

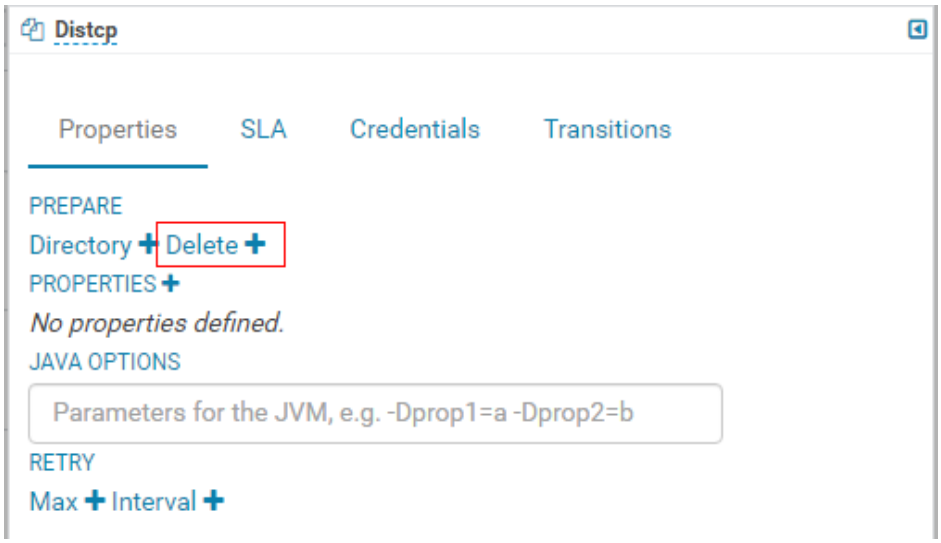
Step 6 Click the configuration button  in the upper right corner. On the **Properties** tab page, click **PROPERTIES+**, enter the attribute name **oozie.launcher.mapreduce.job.hdfs-servers** in the text box on the left, enter the attribute value **hdfs://***source_ip:source_port***,hdfs://***target_ip:target_port* in the text box on the right, and go to **Step 8**.

NOTE

source_ip: service address of the HDFS NameNode in the source cluster
source_port: port number of the HDFS NameNode in the source cluster.
target_ip: service address of the HDFS NameNode in the target cluster
target_port: port number of the HDFS NameNode in the target cluster.

Step 7 In the **Distcp** window that is displayed, set the value of **Source**, for example, to **/user/admin/examples/input-data/text/data.txt**. Set **Destination**, for example, to **/user/admin/examples/output-data/distcp-workflow/data.txt**. Click **Add**.

Step 8 Click  in the upper right corner. On the configuration page that is displayed, click **Delete+** and add the directory to be deleted, for example, **/user/admin/examples/output-data/distcp-workflow**.



Step 9 Click  in the upper right corner of the Oozie editor.

If you need to modify the job name before saving the job (default value: **My Workflow**), click the name directly for modification, for example, **Distcp-Workflow**.

Step 10 After the configuration is saved, click  , and submit the job.

After the job is submitted, you can view the related contents of the job, such as the detailed information, logs, and processes, on Hue.

----End

22.2.13 Submitting an Oozie SSH Job Using Hue

Scenario

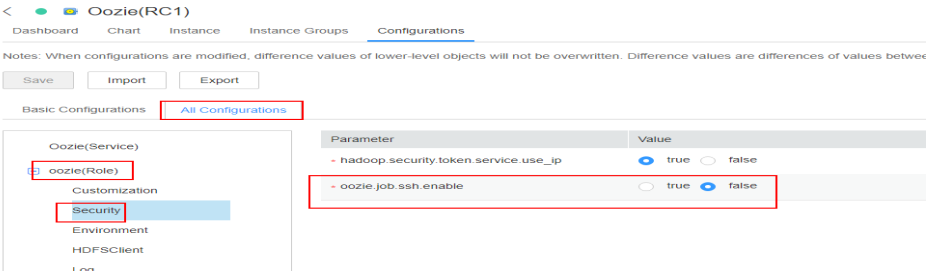
This section guides you to submit an Oozie job of the SSH type on the Hue web UI.

Due to security risks, SSH jobs cannot be submitted by default. To use the SSH function, you need to manually enable it.

Procedure

Step 1 Enable the SSH function. (Skip this step if the **oozie.job.ssh.enable** parameter is unavailable for the current cluster.)

1. On FusionInsight Manager, choose **Cluster** > **Services** > **Oozie** and click the **Configurations** tab and then **All Configurations**. In the navigation pane on the left, choose **oozie(Role)** > **Security**, change the value of **oozie.job.ssh.enable** to **true**, and click **Save**. In the displayed dialog box, click **OK** to save the configuration.



2. On the **Dashboard** page of Oozie, choose **More** > **Restart Service** in the upper-right corner to restart Oozie.

Step 2 Create a workflow. For details, see **Creating a Workflow Using Hue**.

Step 3 For details about how to add the trust relationship, see **Configuring Mutual Trust Between Oozie Nodes**.

Step 4 On the workflow editing page, select the **Ssh** button  and drag it to the operation area.

Step 5 In the **Ssh** window that is displayed, set the following parameters and click **Add**.

- **User and Host**: The value of **User** is the user for which mutual trust is configured in **Step 3**. The parameter value is in the format of *User who performs the SSH task*@*IP address of the node where the SSH task is performed*. For example, the value of this configuration item can be set to **root@***x.x.x.x*.
- **Ssh command**: indicates the command for submitting a job.

Step 6 Click  in the upper right corner of the Oozie editor.

If you need to modify the job name before saving the job (default value: **My Workflow**), click the name directly for modification, for example, **Ssh-Workflow**.

Step 7 After the configuration is saved, click  , and submit the job.

After the job is submitted, you can view the related contents of the job, such as the detailed information, logs, and processes, on Hue.

----End

22.2.14 Submitting a Coordinator Periodic Scheduling Job Using Hue

Scenario

This section describes how to submit a job of the periodic scheduling type on the Hue web UI.

Prerequisites

Required workflow jobs have been configured before the coordinator task is submitted.

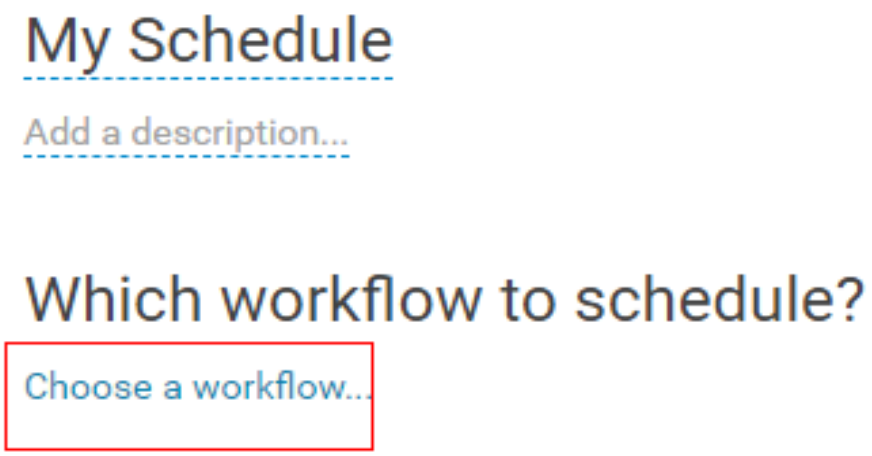
Procedure

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 In the navigation tree on the left, click  and choose **Schedule** to open the Coordinator editor.

Step 3 On the job editing page, click **My Schedule** to change the job name.

Step 4 Click **Choose a workflow...** to select the workflow to be orchestrated.



Step 5 After you select the workflow, set the job execution frequency as prompted. If the workflow to be executed needs to transfer parameters, click **+ Add parameter** to set parameters and click in the upper right corner to save the job.

NOTE

The time may be several hours different from the local time due to time zone conversion.

Step 6 Click in the upper right corner of the editor, set the start value and end value of the time range for executing the scheduled job, and click **Submit** to submit the job.

NOTE

Because the time zone is changed, the difference between the time and the local time may be several hours. For example, in China, the time is 8 hours later than the local time.

----End

22.2.15 Submitting a Bundle Batch Processing Job Using Hue

Scenario

In the case that multiple scheduled jobs exist at the same time, you can manage the jobs in batches over the Bundle task. This section describes how to submit a job of the batch type on the Hue web UI.

Prerequisites

Required related workflow and Coordinator jobs have been configured before the Bundle batch processing job is submitted.

Procedure

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

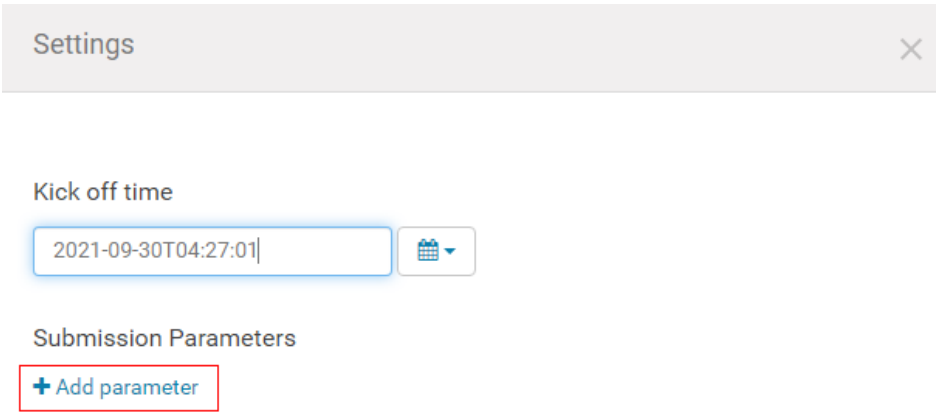
Step 2 In the navigation tree on the left, click and choose **Bundle** to open the Bundle editor.

Step 3 On the job editing page, click **My Bundle** to change the job name.

Step 4 Click **+Add a coordinator** to select the Coordinator job to be orchestrated.

Step 5 Set the start time and the end time for the scheduled coordinator jobs as prompted and click in the upper right corner to save the job.

Step 6 Click in the upper right corner of the editor, select from the displayed menu, set the start time of the bundle task, click **+Add parameter** to add parameters, and close the dialog box to save the settings.



NOTE

Because the time zone is changed, the difference between the time and the local time may be several hours. For example, in China, the time is 8 hours later than the local time.

Step 7 Click in the upper right corner of the editor. In the dialog box that is displayed, click **Submit** to submit the job.

----End

22.2.16 Querying Oozie Job Results on the Hue Page

Scenario

After the jobs are submitted, you can view the execution status of a specific job on Hue.

Procedure

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 Click . On the displayed page, you can view information about the Workflow, Schedule, and Bundle tasks.

View the jobs in the current cluster.

NOTE

The number on **Job Browser** indicates the total number of jobs in the cluster.

Job Browser displays the following job information:

Table 22-7 MRS job attributes

Attribute	Description
Name	Job name
User	User who starts a job
Type	Job type
Status	Job status, including **Succeeded**, **Running**, and **Failed**.
Progress	Job running progress
Group	Group to which a job belongs
Start	Start time of a job
Duration	Job running duration
Id	Job ID, which is generated by the system automatically.

NOTE

If the MRS cluster has Spark, the **Spark-JDBCServer** job is started by default to execute tasks.

----End

22.2.17 Configuring Mutual Trust Between Oozie Nodes

Scenario

This section guides you to enable unidirectional password-free mutual trust when Oozie nodes are used to execute shell scripts of external nodes through SSH jobs.

Prerequisites

You have installed Oozie, and it can communicate with external nodes (nodes connected using SSH).

Procedure

Step 1 Ensure that the user used for SSH connection exists on the external node, and the user directory **~/.ssh** exists.

Step 2 Log in to the node where Oozie is located as user **omm** and check whether the **~/.ssh/id_rsa.pub** file exists.

- If yes, go to **Step 3**.
- If no, run the following command to generate a public-private key pair:

  ssh-keygen -t rsa

Step 3 Log in to the node where the Oozie instance resides as user **omm** and run the following command to configure mutual trust:

ssh-copy-id -i ~/.ssh/id_rsa.pub *User who runs SSH tasks*@*IP address of the node where SSH tasks run*

You need to enter the password of the user who runs SSH tasks as prompted.

NOTE

- The user of the node where Shell resides (external node) must have the permission to execute shell scripts and access all directories and files involved in the Shell scripts.
- If there are multiple Oozie nodes, perform **Step 2** to **Step 3** on all Oozie nodes.

Step 4 Log in to other Oozie nodes as user **omm** and repeat **Step 2** to **Step 3**.

----End

22.3 Enterprise-Class Enhancements of Oozie

22.3.1 Configuring Oozie High Availability (HA)

Scenario

When multiple Oozie nodes provide services at the same time, you can use ZooKeeper to provide high availability (HA), which helps avoid single points of failure (SPOFs) and prevent multiple nodes from concurrently processing the same task.

NOTE

For MRS 3.3.1 or later, the HA mechanism is enabled for Oozie by default. Therefore, you do not need to perform the operations in this section.

Impact on the System

Enabling Oozie HA requires an Oozie restart, and Oozie cannot provide services during the restart.

Prerequisites

- Oozie and ZooKeeper have been installed and are running properly.
- No task is running.
- The current cluster is of the latest version. If it is not, copy the **curator-x-discovery-*x.x.x*.jar** package from the **$BIGDATA_HOME/FusionInsight_Porter_*x.x.x*/install/FusionInsight-Oozie-*x.x.x*/oozie-*x.x.x*/embedded-oozie-server/webapp/WEB-INF/lib** directory to the **$BIGDATA_HOME/FusionInsight_Porter_*x.x.x*/install/FusionInsight-Oozie-*x.x.x*/oozie-*x.x.x*/lib** directory.

Procedure

Step 1 On **FusionInsight Manager**, choose **Cluster** > **Services** > **Oozie**. On the displayed page, click the **Configurations** tab and then **All Configurations**. In the navigation pane on the left, choose **oozie(Role)** > **Customization** and add the configuration items listed in the following table for **oozie.site.configs**. Click **Save** after the modification. In the displayed dialog box, click **OK**.

Parameter	Setting	Description
oozie.services.ext	org.apache.oozie.service.ZKLocksService,org.apache.oozie.service.ZKXLogStreamingService,org.apache.oozie.service.ZKJobsConcurrencyService,org.apache.oozie.service.ZKUUIDService	Services providing enhanced HA
oozie.zookeeper.connection.string	*ZooKeeper instance service IP address:Port number*. Use commas (,) to separate multiple IP address:port pairs.	ZooKeeper connection information
oozie.zookeeper.namespace	oozie	Oozie path on ZooKeeper
oozie.zookeeper.secure	Security cluster: true Normal cluster: not required	Whether to enable Kerberos on ZooKeeper.

Step 2 On the **Dashboard** page of Oozie, choose **More** > **Restart Service** in the upper-right corner to restart Oozie.

----End

22.3.2 Checking Whether the JAR Package on Which Oozie Depends Is Correct Using Share Lib

Oozie tasks require native ShareLib JAR packages to run. ShareLib is automatically uploaded to the **/user/oozie** directory of HDFS when the Oozie kernel is started. Oozie tasks may fail if ShareLib JAR packages in HDFS are damaged, missing, or conflict.

If an Oozie job submitted by a user fails to run, check ShareLib by referring to the operations provided in this section.

This operation applies to MRS 3.3.0 or later.

Prerequisites

- You have installed the HDFS and Oozie clients.
- To check Spark ShareLib, you need to install the Spark client on the node where the Oozie client is located.
- The user who performs the check must have the common user permission of Oozie and the permission to access the **/user/oozie** directory of HDFS.

Procedure

Step 1 Log in to the node where the client is installed as the client installation user.

Step 2 Run the following command to go to the client installation directory:

cd *Client installation directory*

Step 3 Run the following commands to configure environment variables and authenticate the user:

source bigdata_env

kinit *User who submits Oozie tasks* (Skip this step for normal clusters.)

Step 4 Check ShareLib by checking the client or server. Spark ShareLib can be checked only by checking the client.

- Checking the client:
  - Check Oozie ShareLib and ensure that an Oozie instance is installed on the node where the Oozie client to be checked resides.

    oozie -validatesharelib -oozie.core.path=*Oozie instance installation path*

    The following is an example:

    oozie -validatesharelib -oozie.core.path=${BIGDATA_HOME}/FusionInsight_Porter_*/install/FusionInsight-Oozie-*/oozie-*
  - Check Spark ShareLib.

    oozie -validatesharelib -spark.client.path=*Spark client installation directory*

    The following is an example:

    oozie -validatesharelib -spark.client.path=/opt/client/Spark/
- Checking the server:

  Run the following command to check Oozie ShareLib:

  oozie job -oozie https://*Host name of the Oozie role***:21003/oozie -validatesharelib**

  To view the host name of the oozie role, choose **Cluster** > **Services** > **Oozie** and click the **Instance** tab on FusionInsight Manager.

  21003 is the running port of Oozie HTTPS requests. To view the port, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Oozie** and click the **Configuration** tab. Search for **OOZIE_HTTPS_PORT**.

Step 5 View check results. The following circumstances are included:

- Some ShareLib JAR packages are missing.

  If some JAR packages are missing, message "Share Lib jar file(s) not found on hdfs:" and information about the missing JAR packages are displayed.

  If no ShareLib JAR package is missing, message "All Share Lib jar file(s) found on hdfs." is displayed.
- Some JAR packages are damaged.

  If damaged JAR packages are detected, message "Share Lib jar file(s) mismatch on hdfs:" and information about the damaged JAR packages are displayed.

If no ShareLib JAR package is damaged, message "All Share Lib jar file(s) on hdfs match." is displayed.

- Custom JAR packages are uploaded.

  If custom JAR packages are detected, message "Extra Share Lib jar file(s) found on hdfs:" and information about the custom JAR packages are displayed.

  If no custom JAR package is detected, message "No extra Share Lib jar file(s) found on hdfs." is displayed.

Step 6 Rectify the fault based on the check results.

If the detection result obtained in **Step 5** contains information indicating JAR packages are missing or damaged, perform the following operations:

- Spark ShareLib:

  Upload the Spark JAR package in the *Spark client installation directory*/**spark/jars** directory to the HDFS path in the check result.

  hdfs dfs -put -f *Local JAR package path HDFS path that Spark JAR packages are missing or damaged*

- Oozie ShareLib:

  a. Decompress the **oozie-sharelib-*.tar.gz** file under Oozie installation path **${BIGDATA_HOME}/FusionInsight_Porter_*/install/FusionInsight-Oozie-*/oozie-*/** and find the ShareLib JAR package.

     tar -zxf oozie-sharelib-*.tar.gz

  b. Upload the obtained Oozie JAR package to the HDFS path in the check result.

     hdfs dfs -put -f *Local JAR package path HDFS path that Oozie JAR packages are missing or damaged*

----End

22.4 Oozie Log Overview

Log Description

Log path: The default storage paths of Oozie log files are as follows:

- Run log: **/var/log/Bigdata/oozie**
- Audit log: **/var/log/Bigdata/audit/oozie**

Log archiving rule: Oozie logs are classified into run logs, script logs, and audit logs. The maximum size of a run log file is 20 MB, and a maximum of 20 run log files can be reserved. The maximum size of an audit log file is 20 MB, and a maximum of 20 audit log files can be reserved.

NOTE

A compressed log file is generated for **oozie.log** every hour. 720 compressed files (log files of one month) are retained by default.

Table 22-8 Oozie log list

Log Type	Log File Name	Description
Run log	jetty.log	Oozie built-in jetty server log file, which is used to process the request and response information of OozieServlet
	jetty.out	Oozie process startup log file
	oozie_db_temp.log	Oozie database connection log
	oozie-instrumentation.log	Oozie dashboard log file, which records the Oozie running status and configuration information of each component
	oozie-jpa.log	openJPa run log file
	oozie.log	Oozie run log file
	oozie-<SSH_USER>-<DATE>-<PID>-gc.log	Log file that records the garbage collection of the Oozie service
	oozie-ops.log	Oozie operation log file
	check-serviceDetail.log	Oozie health check logs
	oozie-error.log	Oozie running error logs
	threadDump-<DATE>.log	Log file that records stack information when the service process exits normally
Script logs	postinstallDetail.log	Work log file generated after the installation and before the startup
	prestartDetail.log	Pre-startup log file
	startDetail.log	Service startup log file
	stopDetail.log	Service stop log file
	upload-sharelib.log	Operation logs uploaded by **sharelib**
Audit log	oozie-audit.log	Audit log

Log Level

Table 22-9 describes the log levels provided by Oozie.

The priorities of log levels are ERROR, WARN, INFO, and DEBUG in descending order. Logs whose levels are higher than or equal to the set level are printed. The number of printed logs decreases as the configured log level increases.

Table 22-9 Log levels

Level	Description
ERROR	Logs of this level record abnormal information about events that cause process exceptions.
WARN	Logs of this level record exception information about the current event processing.
INFO	Logs of this level record normal running status information about the system and events.
DEBUG	Logs of this level record system information and information about database underlying data transmission.

To modify log levels, perform the following operations:

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > *Name of the desired cluster* > **Services** > **Oozie** > **Configurations**.

Step 3 Select **All Configurations**.

Step 4 On the menu bar on the left, select the log menu of the target role.

Step 5 Select a desired log level.

Step 6 Click **Save**, and then click **OK**. The settings take effect after the processing is complete.

----End

Log Formats

The following table lists the Oozie log formats.

Table 22-10 Log formats

Log Type	Format	Example
Run log	*<yyyy-MM-dd HH:mm:ss,SSS><Log level><Location where the log event occurs><Log level><Message in the log>*	2015-05-29 21:01:45,268 INFO StatusTransitService$StatusTransitRun-nable:539 - USER[-] GROUP[-] Released lock for [org.apache.oozie.service.StatusTransitService]
Script logs	*<yyyy-MM-dd HH:mm:ss,SSS><Host name > <Log level > <Message in the log>*	2015-06-01 17:18:03 001 suse11-192-168-0-111 oozie INFO Running oozie service check script

Log Type	Format	Example
Audit log	*<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|< Thread name >	Message in the log	Location where the log event occurs*	2015-06-01 22:38:41,323 | INFO | http-bio-21003-exec-8 | IP [192.168.0.111] USER [null], GROUP [null], APP [null], JOBID [null], OPERATION [null], PARAMETER [null], RESULT [SUCCESS], HTTPCODE [200], ERRORCODE [null], ERRORMESSAGE [null] | org.apache.oozie.util.XLog.log(XLog.java:539)

22.5 Common Issues About Oozie

22.5.1 Oozie Scheduled Tasks Are Not Executed on Time

Question

Why are not Coordinator scheduled jobs executed on time on the Hue or Oozie client?

Answer

You need to use the UTC time when setting a task.

For example, set **start=2016-12-20T09:00Z** in **job.properties** file.

Modify the configuration and restart the scheduled task.

22.5.2 Why Update of the share lib Directory of Oozie on HDFS Does Not Take Effect?

Symptom

A new JAR package is uploaded to the **/user/oozie/share/lib** directory on HDFS. However, an error indicating that the class cannot be found is reported during task execution.

Solution

Run the following command on the client to refresh the directory:

oozie admin -oozie https://xxx.xxx.xxx.xxx:21003/oozie -sharelibupdate

22.5.3 Common Oozie Troubleshooting Methods

1. Check the job logs on Yarn. Run the command executed through Hive SQL using beeline to ensure that Hive is running properly.

2. If error information such as "classnotfoundException" is displayed, check whether the JAR package of the faulty class exists in the **/user/oozie/share/lib** directory of each component. If no, add the JAR package and go to **Why Update of the share lib Directory of Oozie on HDFS Does Not Take Effect?**. If the faulty class still cannot be found after the **share lib** directory is updated, check whether **sharelibDirNew** is **/user/oozie/share/lib** in the output of the command for updating the directory.

```
[root@host-          client]#
[root@host-          client]#
[root@host-          client]# oozie admin -oozie https://h          :21003/oozie/ -sharelibupdate
INFO CMD=admin -oozie https://host          21003/oozie/ -sharelibupdate
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/client/Oozie/oozie-client-5.1.0-hw-ei-313001-SNAPSHOT/lib/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/client/Oozie/oozie-client-5.1.0-hw-ei-313001-SNAPSHOT/lib/slf4j-simple-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
[ShareLib update status]
        sharelibDirOld = /user/oozie/share/lib
        host = https://          0:21003/oozie
        sharelibDirNew = /user/oozie/share/lib
        status = Successful
```

3. If "NosuchMethodError" is displayed, check whether the JAR packages of each component in the **/user/oozie/share/lib** directory have multiple versions. Note that the JAR packages uploaded by the service cannot conflict with each other. You can check whether a JAR package conflict occurs based on the loaded JAR packages in Oozie run logs on Yarn.
4. If the self-developed code is abnormal, run the Oozie sample to check whether Oozie is running properly.
5. Contact technical support personnel. By using this method, you must collect run logs of Oozie on Yarn, Oozie logs, and component run logs. For example, if an exception occurs when Hive runs on Oozie, you need to collect Hive logs.

23 Using Ranger

23.1 Enabling Ranger Authentication for MRS Cluster Services

Scenario

This section guides you how to enable Ranger authentication. Ranger authentication is enabled by default in security mode and disabled by default in normal mode.

Procedure

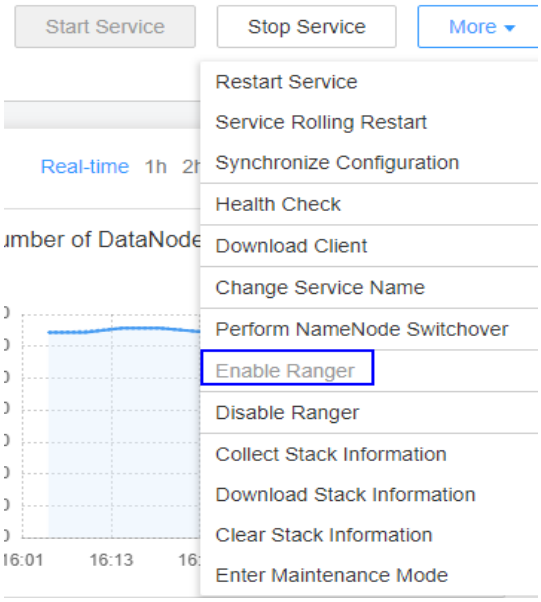
Step 1 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > *Name of the service for which Ranger authentication is enabled*.

Step 2 In the upper right corner of the **Dashboard** page, click **More** and select **Enable Ranger**. In the displayed dialog box, enter the password and click **OK**. After the operation is successful, click **Finish**.

NOTE

- If **Enable Ranger** is dimmed, Ranger authentication is enabled. See **Figure 23-1**.
- For components (except HDFS and YARN) for which Ranger authorization has been enabled, the permissions of non-default roles on Manager do not take effect. You need to configure Ranger policies to assign permissions to user groups.

Figure 23-1 Enabling Ranger Authentication



Step 3 Perform a rolling service restart or restart the service.

----End

23.2 Logging In to the Ranger Web UI

Ranger provides a centralized permission management framework to implement fine-grained permission access control on components, such as HDFS, HBase, Hive, and YARN, and provides a web UI for Ranger administrators to perform operations.

Ranger User Type

Ranger users are classified into **admin**, **user**, and **auditor**. Different users have different permissions to view and operate the Ranger management interface.

- **Admin**: A Ranger security administrator who can view all page content, manage permission management plug-ins and access control policies, view audit information, and set user types.
- **Auditor**: A Ranger audit administrator who can view the permission management plug-ins and access control policies.
- **User**: A common user who can be assigned with specific permissions by the Ranger administrator.

Logging In to the Ranger Web UI

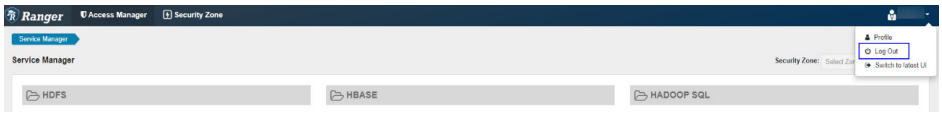
Security mode (Kerberos authentication is enabled for clusters)

Step 1 Log in to FusionInsight Manager as user **admin**. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **Ranger**. The Ranger service overview page is displayed.

Step 2 Click **RangerAdmin** in the **Basic Information** area. The Ranger web UI is displayed.

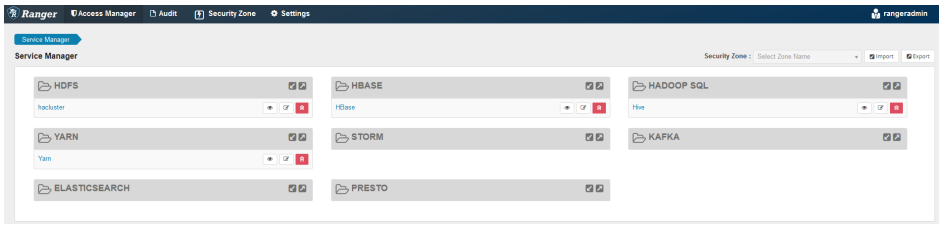
- The **admin** user in Ranger belongs to the **User** type and can only view the **Access Manager** as well as **Security Zone** pages.
- To view all management pages, switch to user **rangeradmin** or other users who have the Ranger administrator permissions.

  a. On the Ranger WebUI, click the user name in the upper right corner and choose **Log Out** to log out of the Ranger WebUI.

  b. Log in to the system as user **rangeradmin** (default password: **Rangeradmin@123**) or another user who has the Ranger administrator permissions. For details about the usernames and default passwords, see User User Account List.

Figure 23-2 Ranger web UI



----End

Normal mode (Kerberos authentication is disabled for clusters)

Step 1 Log in to FusionInsight Manager as user **admin**. For details, see **Accessing FusionInsight Manager**. Choose **Cluster** > **Services** > **Ranger**. The Ranger service overview page is displayed.

Step 2 Click **RangerAdmin** in the **Basic Information** area. The Ranger web UI is displayed.

The **admin** user in Ranger belongs to the **Admin** type and can view all management pages of Ranger without switching to user **rangeradmin**.

NOTE

When a user logs in to the Ranger WebUI as user **rangeradmin** in normal mode, error 401 is reported.

----End

On the homepage of Ranger web UI, you can view the permission management plug-ins of the services integrated in Ranger. The plug-ins can be used to set more fine-grained permissions. For details about functions of main operations you can perform on the page, see **Table 23-1**.

Table 23-1 Functions of each operation portal on the Ranger page

Portal	Function
Access Manager	You can view the permission management plug-ins of each service integrated in Ranger. The plug-ins can be used to set more fine-grained permissions. For details, see **Adding a Ranger Permission Policy**.
Audit	You can view the audit logs related to Ranger running and permission control. For details, see **Viewing Ranger Audit Information**.
Security Zone	Ranger administrators can divide resources of each component into multiple security zones where different Ranger administrators set security policies for specified resources of services to facilitate management. For details, see **Configuring Ranger Security Zone**.
Settings	You can view Ranger permission settings, such as users, user groups, and roles. For details, see **Viewing Ranger User Permission Synchronization Information**.

23.3 Adding a Ranger Permission Policy

By default, Ranger is installed in the newly set up MRS cluster and its authentication model is enabled. The Ranger administrator has the ability to establish detailed security policies for accessing component resources using the component permission plug-ins.

Ranger is currently supported by the following components in clusters with Kerberos authentication enabled: CDL, HDFS, YARN, HBase, Hive, Spark2x, Kafka, Elasticsearch, and HetuEngine.

Configuring User Permission Policies Using Ranger

Step 1 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

Step 2 In the **Service Manager** area on the Ranger homepage, click the permission plug-in name of a component. The page for security access policy list of the component is displayed.

NOTE

In the policy list of each component, many items are generated by default to ensure the permissions of some default users or user groups (such as the **supergroup** user group). Do not delete these items. Otherwise, the permissions of the default users or user groups are affected.

Step 3 Click **Add New Policy** and configure resource access policies for related users or user groups based on the service scenario plan.

The following policies are examples for different components:

- Adding a Ranger Access Permission Policy for CDL
- Adding a Ranger Access Permission Policy for HDFS
- Adding a Ranger Access Permission Policy for HBase
- Adding a Ranger Access Permission Policy for Hive
- Adding a Ranger Access Permission Policy for Yarn
- Adding a Ranger Access Permission Policy for Spark2x
- Adding a Ranger Access Permission Policy for Kafka
- Adding a Ranger Access Permission Policy for HetuEngine

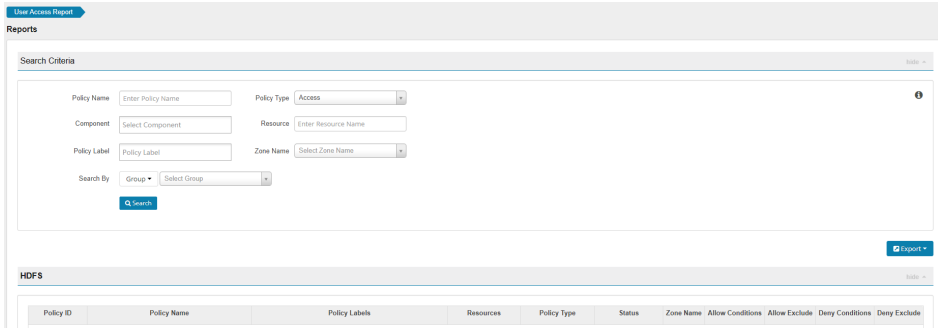
After the policies are added, wait for about 30 seconds for them to take effect.

NOTE

Each time a component is started, the system checks whether the default Ranger service of the component exists. If the service does not exist, the system creates the Ranger service and adds a default policy for it. If a service is deleted by mistake, you can restart or restart the corresponding component service in rolling mode to restore the service. If the default policy is deleted by mistake, you can manually delete the service and then restart the component service.

Step 4 Choose **Access Manager** > **Reports** to view all security access policies of each component.

If there are many system policies, filter and search for policies by the policy name, policy type, component, resource, policy label, security zone, user, or user group. Alternatively, click **Export** to export related policies.



NOTE

- Generally, only one policy can be configured for a fixed resource object. If multiple policies are configured for the same resource object, the policies cannot be saved.
- For details about the priorities of different policies, see **Condition Priorities of the Ranger Permission Policy**.

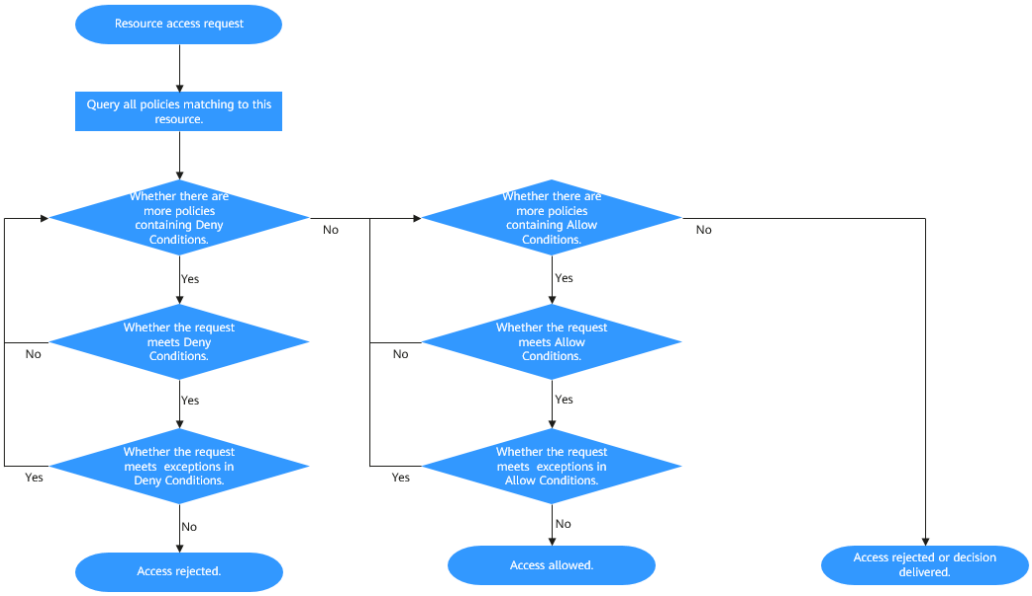
----End

Condition Priorities of the Ranger Permission Policy

When configuring a permission policy for a resource, you can configure Allow Conditions, Exclude from Allow Conditions, Deny Conditions, and Exclude from Deny Conditions for the resource, to meet unexpected requirements in different scenarios.

The priorities of different conditions are listed in descending order: Exclude from Deny Conditions > Deny Conditions > Exclude from Allow Conditions > Allow Conditions

The following figure shows the process of determining condition priorities. If the component resource request does not match the permission policy in Ranger, the system rejects the access by default. However, for HDFS and YARN, the system delivers the decision to the access control layer of the component for determination.



For example, if you want to grant the read and write permissions of the **FileA** folder to the **groupA** user group, but the user in the group is not **UserA**, you can add an allowed condition and an exception condition.

23.4 Configuration Examples for Ranger Permission Policy

23.4.1 Adding a Ranger Access Permission Policy for CDL

Scenario

Ranger administrators can use Ranger to configure creation, execution, query, and deletion permissions for CDL users.

Prerequisites

- The Ranger service has been installed and is running properly.
- You have created users, user groups, or roles for which you want to configure permissions.

Procedure

Step 1 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

Step 2 On the home page, click the component plug-in name in the **CDL** area, for example, **CDL**.

Step 3 Click **Add New Policy** to add a CDL permission control policy.

Step 4 Configure the parameters listed in the table below based on the service demands.

Table 23-2 CDL permission parameters

Parameter	Description
Policy Type	Access.
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**, **192.168.1.20**, or **192.168.1.***.
Policy Name	Policy name, which can be customized and must be unique in the service.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
job	Name of the job applicable to the current policy. You can enter multiple values. The value can contain wildcards, such as **test**, **test***, and *. The **Include** policy applies to the current input object, and the **Exclude** policy applies to objects other than the current input object.
Description	Policy description.
Audit Logging	Whether to audit the policy.

Parameter	Description
Allow Conditions	Permission and exception conditions allowed by a policy. The priority of an exception condition is higher than that of a normal condition. In the **Select Role**, **Select Group**, and **Select User** columns, select the role, user group, or user to which you want to assign permissions. Click **Add Conditions**, add the IP address range to which the policy applies, and click **Add Permissions** to add corresponding permissions. • **Create** permission. • **Execute** permission. • **Delete** permission. • **Update** permission. • **Get** permission. • **Select/Deselect All** permission. To add multiple permission control rules, click  . If users or user groups in the current condition need to manage this policy, select **Delegate Admin**. These users will become the agent administrators. The agent administrators can update and delete this policy and create sub-policies based on the original policy.
Deny Conditions	Policy rejection condition, which is used to configure the permissions and exceptions to be denied in the policy. The configuration method is the same as that of **Allow Conditions**. The priority of the rejection condition is higher than that of the allowed conditions configured in **Allow Conditions**.

Table 23-3 Setting user permissions

Scenario	Role Authorization
Setting the CDL administrator permission	1. On the home page, click the component plug-in name in the **CDL** area, for example, **CDL**. 2. Select the policies whose **Policy Name** is **all - job**, **all - link**, **all - driver** or **all - env**, and click  to edit the policies. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **Select/ Deselect All**.

Scenario	Role Authorization
Setting the permission to manage a CDL job	1. Select a CDL job name from the **job** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Select/ Deselect All**.
Setting the permission to create a CDL job	1. Select a CDL job name from the **job** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Create**. **NOTE** By default, all users have the permission to create a CDL job.
Setting the permission to delete a CDL job	1. Select a CDL job name from the **job** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Delete**.
Setting the permission to obtain information about a CDL job	1. Select a CDL job name from the **job** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Get**.
Setting the permission to execute a CDL job	1. Select a CDL job name from the **job** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Execute**.
Setting the permission to manage a CDL data link	1. Select the CDL data link name on the right of the **link** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Select/ Deselect All**.

Scenario	Role Authorization
Setting the permission to create a CDL data link	1. Select the CDL data link name on the right of the **link** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Create**. **NOTE** By default, all users have the permission to creat CDL data links.
Setting the permission to delete a CDL data link	1. Select the CDL data link name on the right of the **link** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Delete**.
Setting the permission to update a CDL data link	1. Select the CDL data link name on the right of the **link** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Update**.
Setting the permission to obtain information about a CDL data link	1. Select the CDL data link name on the right of the **link** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Get**.
Setting the permission to manage a CDL driver	1. Select the CDL driver name on the right of the **driver** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Select/ Deselect All**.
Setting the permission to delete a CDL driver	1. Select the CDL driver name on the right of the **driver** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Delete**.
Setting the permission to update a CDL driver	1. Select the CDL driver name on the right of the **driver** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Update**.

Scenario	Role Authorization
Setting the permission to obtain information about a CDL driver	1. Select the CDL driver name on the right of the **driver** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Get**.
Setting the permission to manage a CDL environment variable	1. Select the CDL environment variable name on the right of the **env** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Select/ Deselect All**.
Setting the permission to create a CDL environment variable	1. Select the CDL environment variable name on the right of the **env** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Create**. **NOTE** By default, all users have the permission to create a CDL environment variable.
Setting the permission to delete a CDL environment variable	1. Select the CDL environment variable name on the right of the **env** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Delete**.
Setting the permission to update a CDL environment variable	1. Select the CDL environment variable name on the right of the **env** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Update**.
Setting the permission to obtain information about a CDL environment variable	1. Select the CDL environment variable name on the right of the **env** drop-down list. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Get**.

Step 5 (Optional) Add the validity period of the policy. Click **Add Validity period** in the upper right corner of the page, set **Start Time** and **End Time**, and select **Time Zone**. Click **Save**. To add multiple policy validity periods, click  . To delete a policy validity period, click  .

Step 6 Click **Add** to view the basic information about the policy in the policy list. After the policy takes effect, check whether the related permissions are normal.

To disable a policy, click  to edit the policy and set the policy to **Disabled**.

If a policy is no longer used, click  to delete it.

----End

23.4.2 Adding a Ranger Access Permission Policy for HDFS

Scenario

Ranger administrators can use Ranger to configure the read, write, and execution permissions on HDFS directories or files for HDFS users.

Prerequisites

- The Ranger service has been installed and is running properly.
- You have created users, user groups, or roles for which you want to configure permissions.

Procedure

Step 1 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

Step 2 On the homepage, click the component plug-in name in the **HDFS** area, for example, **hacluster**.

Step 3 Click **Add New Policy** to add an HDFS permission control policy.

Step 4 Configure the parameters listed in the table below based on the service demands.

Table 23-4 HDFS permission parameters

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service.
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**,**192.168.1.20**, or **192.168.1.***.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.

Parameter	Description
Resource Path	Resource path, which is the HDFS path folder or file to which the current policy applies. You can enter multiple values and use the wildcard (*), for example, **/test/***. To enable a subdirectory to inherit the permission of its upper-level directory, enable the recursion function. If recursion is enabled for the parent directory and a policy is configured for the subdirectory, the policy configured for the subdirectory is used. • **non-recursive**: recursion disabled • **recursive**: recursion enabled
Description	Policy description.
Audit Logging	Whether to audit the policy.
Allow Conditions	Permission and exception conditions allowed by a policy. The priority of an exception condition is higher than that of a normal condition. In the **Select Role**, **Select Group**, and **Select User** columns, select the role, user group, or user to which the permission is to be granted, click **Add Conditions**, add the IP address range to which the policy applies, and click **Add Permissions** to add the corresponding permission. • **Read**: permission to read data • **Write**: permission to write data • **Execute**: execution permission • **Select/Deselect All**: Select or deselect all. If users or user groups in the current condition need to manage this policy, select **Delegate Admin**. These users or user groups will become the agent administrators. The agent administrators can update and delete this policy and create sub-policies based on the original policy. To add multiple permission control rules, click  . To delete a permission control rule, click  . Exclude from Allow Conditions: exception rules excluded from the allowed conditions
Deny All Other Accesses	Whether to reject all other access requests. • **True**: All other access requests are rejected. • **False**: **Deny Conditions** can be configured.

Parameter	Description
Deny Conditions	Policy rejection condition, which is used to configure the permissions and exceptions to be denied in the policy. The configuration method is the same as that of **Allow Conditions**. The priority of the rejection condition is higher than that of the allowed conditions configured in **Allow Conditions**. **Exclude from Deny Conditions**: exception rules excluded from the denied conditions

For example, to add the write permission for the **/user/test** directory of user **testuser**, the configuration is as follows:

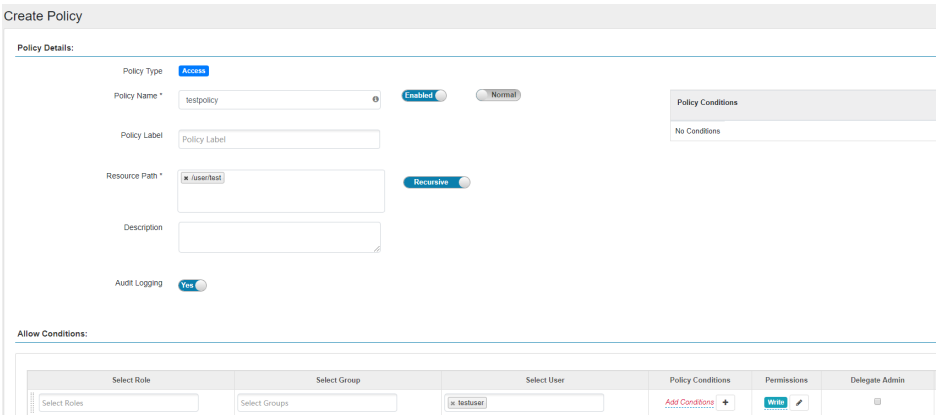


Table 23-5 Setting permissions

Task	Role Authorization
Setting the HDFS administrator permission	1. On the homepage, click the component plug-in name in the **HDFS** area, for example, **hacluster**. 2. Select the policy whose **Policy Name** is **all - path** and click  to edit the policy. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list.
Setting the permission for users to check and recover HDFS	1. Add a folder or a file path in **Resource Path**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Read** and **Execute**.

Task	Role Authorization
Setting the permission for users to read directories or files of other users	1. Add a folder or a file path in **Resource Path**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Read** and **Execute**.
Setting the permission for users to write data to files of other users	1. Add a folder or a file path in **Resource Path**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Write** and **Execute**.
Setting the permission for users to create or delete sub-files or sub-directories in the directory of other users	1. Add a folder or a file path in **Resource Path**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Write** and **Execute**.
Setting the permission for users to execute directories or files of other users	1. Add a folder or a file path in **Resource Path**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Execute**.
Setting the permission for allowing subdirectories to inherit all permissions of their parent directories	1. Add a folder or a file path in **Resource Path**. 2. Enable the recursion function. **Recursive** indicates that recursion is enabled.

Step 5 (Optional) Add the validity period of the policy. Click **Add Validity period** in the upper right corner of the page, set **Start Time** and **End Time**, and select **Time Zone**. Click **Save**. To add multiple policy validity periods, click  . To delete a policy validity period, click  .

Step 6 Click **Add** to view the basic information about the policy in the policy list. After the policy takes effect, check whether the related permissions are normal.

To disable a policy, click  to edit the policy and set the policy to **Disabled**.

If a policy is no longer used, click  to delete it.

----End

23.4.3 Adding a Ranger Access Permission Policy for HBase

Scenario

Ranger administrators can use Ranger to configure permissions on HBase tables, column families, and columns for HBase users.

Prerequisites

- The Ranger service has been installed and is running properly.
- You have created users, user groups, or roles for which you want to configure permissions.

Procedure

Step 1 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

Step 2 On the home page, click the component plug-in name in the **HBASE** area, for example, **HBase**.

Step 3 Click **Add New Policy** to add an HBase permission control policy.

Step 4 Configure the parameters listed in the table below based on the service demands.

Table 23-6 HBase permission parameters

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service.
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**,**192.168.1.20**, or **192.168.1.***.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
HBase Table	Name of a table to which the policy applies. The value can contain wildcard (*). For example, **table1:*** indicates all tables in **table1**. The **Include** policy applies to the current input object, and the **Exclude** policy applies to objects other than the current input object. **NOTE** The value of **hbase.rpc.protection** of the HBase service plug-in on Ranger must be the same as that of **hbase.rpc.protection** on the HBase server. For details, see **Existing HBase Tables Cannot Be Searched Using Wildcards When HBase Permission Policies Are Configured**.

Parameter	Description
HBase Column-family	Name of the column families to which the policy applies. The **Include** policy applies to the current input object, and the **Exclude** policy applies to objects other than the current input object.
HBase Column	Name of the column to which the policy applies. The **Include** policy applies to the current input object, and the **Exclude** policy applies to objects other than the current input object.
Description	Policy description.
Audit Logging	Whether to audit the policy.
Allow Conditions	Policy allowed condition. You can configure permissions and exceptions allowed by the policy. In the **Select Role**, **Select Group**, and **Select User** columns, select the role, user group, or user to which the permission is to be granted, click **Add Conditions**, add the IP address range to which the policy applies, and click **Add Permissions** to add the corresponding permission. • **Read**: permission to read data • **Write**: permission to write data • **Create**: permission to create data • **Admin**: permission to manage data • **Select/Deselect All**: Select or deselect all. If users or user groups in the current condition need to manage this policy, select **Delegate Admin**. These users or user groups will become the agent administrators. The agent administrators can update and delete this policy and create sub-policies based on the original policy. To add multiple permission control rules, click  . To delete a permission control rule, click  . Exclude from Allow Conditions: policy exception conditions
Deny All Other Accesses	Whether to reject all other access requests. • **True**: All other access requests are rejected. • **False**: **Deny Conditions** can be configured.

Parameter	Description
Deny Conditions	Policy rejection condition, which is used to configure the permissions and exceptions to be denied in the policy. The configuration method is similar to that of **Allow Conditions**. The priority of **Deny Conditions** is higher than that of allowed conditions configured in **Allow Conditions**. **Exclude from Deny Conditions**: exception rules excluded from the denied conditions

Table 23-7 Setting permissions

Task	Role Authorization
Setting the HBase administrator permission	1. On the home page, click the component plug-in name in the **HBase** area, for example, **HBase**. 2. Select the policy whose **Policy Name** is **all - table, column-family, column** and click  to edit the policy. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list.
Setting the permission for users to create tables	1. In **HBase Table**, specify a table name. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Create**. 4. This user hase the following permissions: create table drop table truncate table alter table enable table flush table flush region compact disable enable desc

Task	Role Authorization
Setting the permission for users to write data to tables	1. In **HBase Table**, specify a table name. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Write**. 4. The user has the **put**, **delete**, **append**, and **incr** operation permissions.
Setting the permission for users to read data from tables	1. In **HBase Table**, specify a table name. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Read**. 4. This user hase the **get** and **scan** permissions.
Setting the permission for users to manage namespaces or tables	1. In **HBase Table**, specify a table name. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Admin**. 4. The user has the **rsgroup**, **peer**, **assign** and **balance** operation permissions.
Setting the permission for reading data from or writing data to columns	1. In **HBase Table**, specify a table name. 2. In **HBase Column-family**, specify the column family name. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **Read** and **Write**.

NOTE

If a user performs the **desc** operation in **hbase shell**, the user must be granted the read permission on the **hbase:quota** table.

Step 5 (Optional) Add the validity period of the policy. Click **Add Validity period** in the upper right corner of the page, set **Start Time** and **End Time**, and select **Time Zone**. Click **Save**. To add multiple policy validity periods, click  . To delete a policy validity period, click  .

Step 6 Click **Add** to view the basic information about the policy in the policy list. After the policy takes effect, check whether the related permissions are normal.

To disable a policy, click  to edit the policy and set the policy to **Disabled**.

If a policy is no longer used, click  to delete it.

----End

23.4.4 Adding a Ranger Access Permission Policy for Hive

Scenario

Ranger administrators can use Ranger to set permissions for Hive users. The default administrator account of Hive is **hive** and the initial password is **Hive@123**.

Prerequisites

- The Ranger service has been installed and is running properly.
- You have created users, user groups, or roles for which you want to configure permissions.
- The users must be added to the **hive** group.

Procedure

Step 1 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

Step 2 On the home page, click the component plug-in name in the **HADOOP SQL** area, for example, **Hive**.

Step 3 On the **Access** tab page, click **Add New Policy** to add a Hive permission control policy.

Step 4 Configure the parameters listed in the table below based on the service demands.

Table 23-8 Hive permission parameters

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service.
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**, **192.168.1.20**, or **192.168.1.***.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
database	Name of the Hive database to which the policy applies. The **Include** policy applies to the current input object, and the **Exclude** policy applies to objects other than the current input object.

Parameter	Description
table	Name of the Hive table to which the policy applies. To add a UDF-based policy, switch to UDF and enter the UDF name. The **Include** policy applies to the current input object, and the **Exclude** policy applies to objects other than the current input object.
Hive Column	Name of the column to which the policy applies. The value ***** indicates all columns. The **Include** policy applies to the current input object, and the **Exclude** policy applies to objects other than the current input object.
Description	Policy description.
Audit Logging	Whether to audit the policy.
Allow Conditions	Policy allowed condition. You can configure permissions and exceptions allowed by the policy. In the **Select Role**, **Select Group**, and **Select User** columns, select the role, user group, or user to which the permission is to be granted, click **Add Conditions**, add the IP address range to which the policy applies, and click **Add Permissions** to add the corresponding permission. ● select: permission to query data ● update: permission to update data ● Create: permission to create data ● Drop: permission to drop data ● Alter: permission to alter data ● Index: permission to index data ● All: all permissions ● Read: permission to read data ● Write: permission to write data ● Temporary UDF Admin: temporary UDF management permission ● Select/Deselect All: Select or deselect all. To add multiple permission control rules, click  . If users or user groups in the current condition need to manage this policy, select **Delegate Admin**. These users will become the agent administrators. The agent administrators can update and delete this policy and create sub-policies based on the original policy.

Parameter	Description
Deny Conditions	Policy rejection condition, which is used to configure the permissions and exceptions to be denied in the policy. The configuration method is similar to that of **Allow Conditions**.

Table 23-9 Setting permissions

Task	Role Authorization
role admin operation	1. On the home page, click **Settings** and choose **Roles**. 2. Click the role with **Role Name** set to **admin**. In the **Users** area, click **Select User** and select a username. 3. Click **Add Users**, select **Is Role Admin** in the row where the username is located, and click **Save**. **NOTE** Only user **rangeradmin** has the permission to access the **Settings** option on the Ranger page. After being bound to the Hive administrator role, perform the following operations during each maintenance operation: 1. Log in to the node where the Hive client is installed as the client installation user. 2. Run the following command to configure environment variables: For example, if the Hive client installation directory is **/opt/hiveclient**, run **source /opt/hiveclient/bigdata_env**. 3. Run the following command to authenticate the user: **kinit** *Hive service user* 4. Run the following command to log in to the client tool: **beeline** 5. Run the following command to update the administrator permissions: **set role admin;**
Creating a database table	1. Enter the policy name in **Policy Name**. 2. Enter or select the corresponding database on the right side of **database** and enter or select ***** on the right side of **column**. (To create a table, enter or select the corresponding table on the right side of **table**.) 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **Create**.

Task	Role Authorization
Deleting a table	1. Enter the policy name in **Policy Name**. 2. Enter or select the corresponding database on the right side of **database** and enter and select ***** on the right side of **column**. (To delete a table, enter or select the corresponding table on the right side of **table**.) 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **Drop**.
Query operation (**select**, **desc**, and **show**)	1. Enter the policy name in **Policy Name**. 2. Enter or select the corresponding database on the right side of **database** and enter or select ***** (***** indicates all columns) on the right side of **column**. (To create a table, enter or select the corresponding table on the right side of **table**.) 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **select**.
Alter operation	1. Enter the policy name in **Policy Name**. 2. Enter and select the corresponding database on the right side of **database** and enter or select ***** on the right side of **column**. (For tables, enter or select the corresponding table on the right side of **table**.) 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **Alter**.
LOAD operation	1. Enter the policy name in **Policy Name**. 2. On the right side of **database**, enter or select the corresponding database. On the right side of **table**, enter or select the corresponding table. On the right side of **column**, enter a column and select *****. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **update**.

Task	Role Authorization
INSERT and **DELETE** operations	1. Enter the policy name in **Policy Name**. 2. On the right side of **database**, enter or select the corresponding database. On the right side of **table**, enter or select the corresponding table. On the right side of **column**, enter a column and select *****. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **update**. 5. Configure the **submit** permission on the Yarn task queue. For details about how to configure the permission, see **Adding a Ranger Access Permission Policy for Yarn**.
Import/Export operation	1. Enter the policy name in **Policy Name**. 2. On the right side of **database**, enter or select the corresponding database. On the right side of **table**, enter or select the corresponding table. On the right side of **column**, enter a column and select *****. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **select**. **NOTE** This feature applies only to MRS 3.2.0 or later.
Repl Dump/Load operation	1. Enter the policy name in **Policy Name**. 2. On the right side of **database**, enter or select the corresponding database. On the right side of **table**, enter a table or select *****. On the right side of **column**, enter a column or select *****. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **ReplAdmin**. **NOTE** This feature applies only to MRS 3.2.0 or later.
GRANT/REVOKE operation	1. Enter the policy name in **Policy Name**. 2. On the right side of **database**, enter or select the corresponding database. On the right side of **table**, enter or select the corresponding table. On the right side of **column**, enter a column and select *****. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Select **Delegate Admin**.

Task	Role Authorization
ADD JAR operation	1. Enter the policy name in **Policy Name**. 2. Click **database**, and select **global** from the drop-down list. On the right of **global**, enter related information or select *****. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **Temporary UDF Admin**.
UDF operation	1. Enter the policy name in **Policy Name**. 2. Enter or select the corresponding database on the right of **database**, and enter the corresponding udf function name on the right of **udf**. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select required permissions for the user (**udf** supports the **Create**, **select**, and **Drop** permissions).
VIEW operation	1. Enter the policy name in **Policy Name**. 2. On the right side of **database**, enter or select the corresponding database. On the right side of **table**, enter or select the corresponding table to be viewed. On the right side of **column**, enter a column and select *****. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select permissions for the user as required.
dfs command operation	The **dfs** operation can be performed only after you have run the **set role admin** command.
Operations on other user database tables	1. Perform the preceding operations to add the corresponding permissions. 2. Grant the read, write, and execution permissions on the HDFS paths of other user database tables to the user. For details, see **Adding a Ranger Access Permission Policy for HDFS**.

NOTE

- If you have specified an HDFS path when running commands, you need to be granted with the read, write, and execution permissions on the HDFS paths. For details, see **Adding a Ranger Access Permission Policy for HDFS**. You do not need to configure the Ranger policy of HDFS. You can use the Hive permission plug-in to add permissions to the role and assign the role to the corresponding user. If the HDFS Ranger policy can match the file or directory permission of the Hive database table, the HDFS Ranger policy is preferentially used.
- MRS 3.3.0 or later: If the cascading authorization function of Hive tables has been enabled by referring to **Hive Tables Supporting Cascading Authorization**, you do not need to authorize the HDFS path where the table is located.
- If the Hive table is stored on OBS, you can use the URL policy within the Ranger policy. Set the URL to the complete path of the object on OBS. The read and write permissions are used together with the URL. URL policies are not involved in other scenarios.
- The global policy in the Ranger policy is used only with the **Temporary UDF Admin** permission to control the upload of UDF packages.
- The **hiveservice** policy in the Ranger policy is used only with the **Service Admin** permission to control the permission to run the **kill query <queryId>** command to end the task that is being executed.
- The **lock**, **index**, **refresh**, and **replAdmin** permissions are not supported.
- Run the **show grant** command to view the table permission. The **grantor** column of the table **owner** is displayed as user **hive**. If the Ranger page is used or the **grant** command is used to grant permissions in the background, the **grantor** column is displayed as the corresponding user. To view the result of using the Hive permission plug-in, set **hive-ext.ranger.previous.privileges.enable** to **true** and run the **show grant** command.

Step 5 Click **Add** to view the basic information about the policy in the policy list. After the policy takes effect, check whether the related permissions are normal.

To disable a policy, click  to edit the policy and set the policy to **Disabled**.

If a policy is no longer used, click  to delete it.

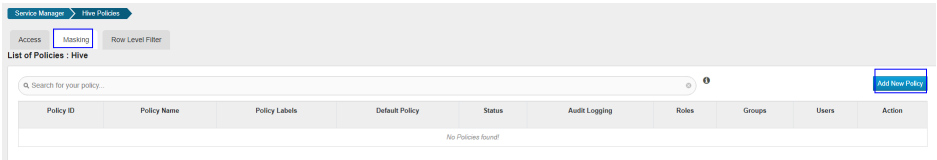
----End

Hive Data Masking

Ranger supports data masking for Hive data. It can process the returned result of the **select** operation you performed to mask sensitive information.

Step 1 Log in to the Ranger web UI. Click **Hive** in the **HADOOP SQL** area on the homepage.

Step 2 On the **Masking** tab page, click **Add New Policy** to add a Hive permission control policy.



Step 3 Configure the parameters listed in the table below based on the service demands.

Table 23-10 Hive data masking parameters

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service.
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**, **192.168.1.20**, or **192.168.1.***.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
Hive Database	• This parameter indicates the name of the Hive database to which the current policy applies. • In MRS 3.3.0 and later versions, multiple database names can be configured and wildcards (*) are supported, for example, **aa**, **a***, ***b**, **a*b**, or ***.
Hive Table	• This parameter indicates the name of the Hive table to which the current policy applies. • In MRS 3.3.0 and later versions, multiple table names can be configured and wildcards (*) are supported, for example, **aa**, **a***, ***b**, **a*b**, or ***.
Hive Column	• Column name addition is supported. • In MRS 3.3.0 and later versions, multiple column names can be configured and wildcards (*) are supported, for example, **aa**, **a***, ***b**, **a*b**, or ***.
Description	Policy description.
Audit Logging	Whether to audit the policy.

Parameter	Description
Mask Conditions	In the **Select Role**, **Select Group**, and **Select User** columns, select the object to which the permission is to be granted, click **Add Conditions**, add the IP address range to which the policy applies, then click **Add Permissions**, and select **select**. Click **Select Masking Option** and select a data masking policy. • Redact: Use **x** to mask all letters and **0** to mask all digits. • Partial mask: show last 4: Only the last four characters are displayed, and the rest characters are displayed using **x**. • Partial mask: show first 4: Only the first four characters are displayed, and the rest characters are displayed using **x**. • Hash: Replace the original value with the hash value. The Hive built-in function **mask_hash** is used. This is valid only for fields of the string, character, and varchar types. NULL is returned for fields of other types. • Nullify: Replace the original value with the NULL value. • Unmasked (retain original value): Keep the original value. • Date: show only year: Only the year part of the date string is displayed, and the default month and date start from January and Monday (**01/01**). • Custom: You customize policies using any valid return data type which is the same as the data type in the masked column. To add a multi-column masking policy, click  .

Step 4 Click **Add** to view the basic information about the policy in the policy list.

Step 5 After you perform the **select** operation on a table configured with a data masking policy on the Hive client, the system processes and displays the data.

NOTE

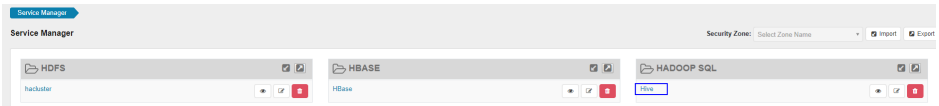
To process data, you must have the permission to submit tasks to the Yarn queue.

----End

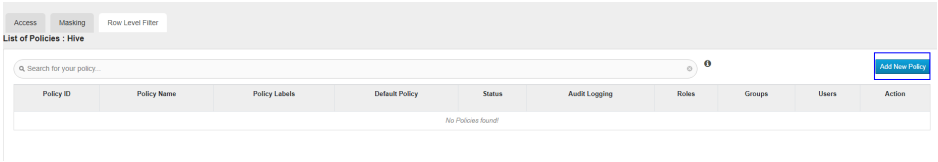
Hive Row-Level Data Filtering

Ranger allows you to filter data at the row level when you perform the **select** operation on Hive data tables.

Step 1 Log in to the Ranger web UI. Click **Hive** in the **HADOOP SQL** area on the homepage.



Step 2 On the **Row Level Filter** tab page, click **Add New Policy** to add a row data filtering policy.



Step 3 Configure the parameters listed in the table below based on the service demands.

Table 23-11 Parameters for filtering Hive row data

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service.
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**, **192.168.1.20**, or **192.168.1.***.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
Hive Database	Name of the Hive database to which the current policy applies.
Hive Table	Name of the Hive table to which the current policy applies.
Description	Policy description.
Audit Logging	Whether to audit the policy.
Row Filter Conditions	In the **Select Role**, **Select Group**, and **Select User** columns, select the object to which the permission is to be granted, click **Add Conditions**, add the IP address range to which the policy applies, then click **Add Permissions**, and select **Select**. Click **Row Level Filter** and enter data filtering rules. For example, if you want to filter the data in the **zhangsan** row in the **name** column of **table A**, the filtering rule is **name <>'zhangsan'**. For more information, see the official Ranger document. To add more rules, click  .

Step 4 Click **Add** to view the basic information about the policy in the policy list.

Step 5 After you perform the **select** operation on a table configured with a data masking policy on the Hive client, the system processes and displays the data.

NOTE

To process data, you must have the permission to submit tasks to the Yarn queue.

----End

23.4.5 Adding a Ranger Access Permission Policy for Yarn

Scenario

Ranger administrators can use Ranger to configure YARN administrator permissions for YARN users, allowing them to manage YARN queue resources.

Prerequisites

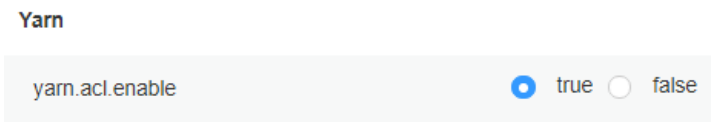
- The Ranger service has been installed and is running properly.
- You have created users, user groups, or roles for which you want to configure permissions.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Yarn**.

Step 2 On the page that is displayed, click the **Configuration** tab then the **All Configurations** sub-tab. On this sub-tab page, search for the **yarn.acl.enable** parameter, and change its value to **true**. If the value is **true**, no further action is required.

Figure 23-3 Configuring yarn.acl.enable



Step 3 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

Step 4 On the home page, click the component plug-in name in the **YARN** area, for example, **Yarn**.

Step 5 Click **Add New Policy** to add a Yarn permission control policy.

Step 6 Configure the parameters listed in the table below based on the service demands.

Table 23-12 Yarn permission parameters

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service.

Parameter	Description
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**,**192.168.1.20**, or **192.168.1.***.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
Queue	Queue name. The wildcard (*) is supported. To enable a sub-queue to inherit the permission of its upper-level queue, enable the recursion function. • **Non-recursive**: recursion disabled • **Recursive**: recursion enabled
Description	Policy description.
Audit Logging	Whether to audit the policy.
Allow Conditions	Policy allowed condition. You can configure permissions and exceptions allowed by the policy. In the **Select Role**, **Select Group**, and **Select User** columns, select the role, user group, or user to which the permission is to be granted, click **Add Conditions**, add the IP address range to which the policy applies, and click **Add Permissions** to add the corresponding permission. • submit-app: permission to submit queue tasks • admin-queue: permission to manage queue tasks • Select/Deselect All: Select or deselect all. If users or user groups in the current condition need to manage this policy, select **Delegate Admin**. These users will become the agent administrators. The agent administrators can update and delete this policy and create sub-policies based on the original policy. To add multiple permission control rules, click  . To delete a permission control rule, click  . Exclude from Allow Conditions: policy exception conditions
Deny All Other Accesses	Whether to reject all other access requests. • **True**: All other access requests are rejected. • **False**: **Deny Conditions** can be configured.

Parameter	Description
Deny Conditions	Policy rejection condition, which is used to configure the permissions and exceptions to be denied in the policy. The configuration method is similar to that of **Allow Conditions**. The priority of **Deny Conditions** is higher than that of allowed conditions configured in **Allow Conditions**. Exclude from Deny Conditions: exception rules excluded from the denied conditions

Table 23-13 Setting permissions

Task	Role Authorization
Setting the Yarn administrator permission	1. On the home page, click the component plug-in name in the **YARN** area, for example, **Yarn**. 2. Select the policy whose **Policy Name** is **all - queue** and click  to edit the policy. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list.
Setting the permission for a user to submit tasks in a specified Yarn queue	1. In **Queue**, specify a queue name. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **submit-app**.
Setting the permission for a user to manage tasks in a specified Yarn queue	1. In **Queue**, specify a queue name. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **admin-queue**.

Step 7 (Optional) Add the validity period of the policy. Click **Add Validity period** in the upper right corner of the page, set **Start Time** and **End Time**, and select **Time Zone**. Click **Save**. To add multiple policy validity periods, click  . To delete a policy validity period, click  .

Step 8 Click **Add** to view the basic information about the policy in the policy list. After the policy takes effect, check whether the related permissions are normal.

To disable a policy, click  to edit the policy and set the policy to **Disabled**.

If a policy is no longer used, click  to delete it.

----End

NOTE

The permissions on Ranger Yarn are independent of each other. There is inclusion relationship among the permissions. Currently, the following permissions are supported:

- **submit-app**: permission to submit queue tasks
- **admin-queue**: permission to manage queue tasks

Although the **admin-queue** has the permission to submit tasks, it does not have the inclusion relationship with the **submit-app** permission.

23.4.6 Adding a Ranger Access Permission Policy for Spark2x

Scenario

Ranger administrators can use Ranger to set permissions for Spark2x users.

NOTE

- After Ranger authentication is enabled or disabled on Spark2x, you need to restart Spark2x.
- Download the client again or manually update client configuration file **spark-defaults.conf** in the *Client installation directory*/**Spark2x/spark/conf** directory.

  To enable Ranger authentication, set **spark.ranger.plugin.authorization.enable** to **true** and change the value of **spark.sql.authorization.enabled** to **true**.

  Disable Ranger: **spark.ranger.plugin.authorization.enable=false**
- Spark2x spark-beeline, which connects to JDBCServer, allows for Ranger IP address filtering policy (**Policy Conditions** in Ranger permission policy), but this feature is not available in spark-submit and spark-sql.
- In MRS 3.3.0-LTS and later versions, the Spark2x component is renamed Spark, and the role names of this component are also changed. For example, JobHistory2x is changed to JobHistory. Refer to the descriptions and operations related to the component name and role names in the document based on your MRS version.

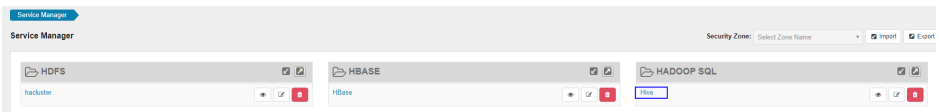
Prerequisites

- The Ranger service has been installed and is running properly.
- Ranger authentication of Hive has been enabled and the Spark Ranger authentication function is reactivated following the sequential reboot of Hive and then Spark. The Spark service is restarted after the Spark Ranger authentication function is reactivated.
- You have created users, user groups, or roles for which you want to configure permissions.
- The created user has been added to the **hive** user group.

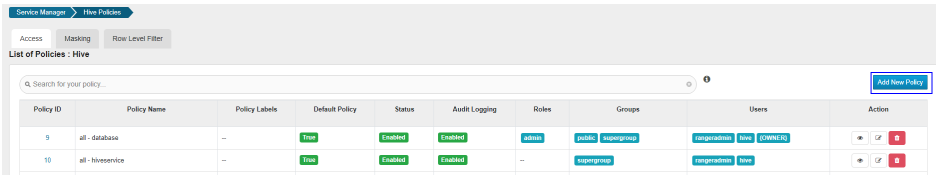
Procedure

Step 1 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

Step 2 On the home page, click the component plug-in name in the **HADOOP SQL** area, for example, **Hive**.



Step 3 On the **Access** tab page, click **Add New Policy** to add a Spark2x permission control policy.



Step 4 Configure the parameters listed in the table below based on the service demands.

Table 23-14 Spark2x permission parameters

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service.
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**,**192.168.1.20**, or **192.168.1.***.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
database	Name of the Spark2x database to which the policy applies. The **Include** policy applies to the current input object, and the **Exclude** policy applies to objects other than the current input object.
table	Name of the Spark2x table to which the policy applies. To add a UDF-based policy, switch to UDF and enter the UDF name. The **Include** policy applies to the current input object, and the **Exclude** policy applies to objects other than the current input object.
column	Name of the column to which the policy applies. The value * indicates all columns. The **Include** policy applies to the current input object, and the **Exclude** policy applies to objects other than the current input object.
Description	Policy description.
Audit Logging	Whether to audit the policy.

Parameter	Description
Allow Conditions	Policy allowed condition. You can configure permissions and exceptions allowed by the policy. In the **Select Role**, **Select Group**, and **Select User** columns, select the role, user group, or user to which the permission is to be granted, click **Add Conditions**, add the IP address range to which the policy applies, and click **Add Permissions** to add the corresponding permission. ● select: permission to query data ● update: permission to update data ● Create: permission to create data ● Drop: permission to drop data ● Alter: permission to alter data ● Index: permission to index data ● All: all permissions ● Read: permission to read data ● Write: permission to write data ● Temporary UDF Admin: temporary UDF management permission ● Select/Deselect All: Select or deselect all. To add multiple permission control rules, click  . If users or user groups in the current condition need to manage this policy, select **Delegate Admin**. These users will become the agent administrators. The agent administrators can update and delete this policy and create sub-policies based on the original policy.
Deny Conditions	Policy rejection condition, which is used to configure the permissions and exceptions to be denied in the policy. The configuration method is similar to that of **Allow Conditions**.

Table 23-15 Setting permissions

Task	Operation
role admin operation	1. On the home page, click **Settings** and choose **Roles** > **Add New Role**. 2. Set **Role Name** to **admin**. In the **Users** area, click **Select User** and select a username. 3. Click **Add Users**, select **Is Role Admin** in the row where the username is located, and click **Save**. **NOTE** After being bound to the Hive administrator role, perform the following operations during each maintenance operation: 1. Log in to the node where the Hive client is installed as the client installation user. 2. Run the following command to configure environment variables: For example, if the installation directory of the Spark2x client is /opt/client, run the **source /opt/client/bigdata_env** command. 3. Run the following command to perform user authentication: **kinit Spark2x***Service user* 4. Run the following command to log in to the client tool: **spark-beeline** 5. Run the following command to update the administrator permissions: **set role admin;**
Creating a database table	1. Enter the policy name in **Policy Name**. 2. Enter and select the corresponding database on the right of **database**. (If you want to create a database, enter the name of the database to be created or enter ***** to indicate a database with any name, and then select the name.) Enter and select the corresponding table name on the right of **table** and **column**. Wildcard characters (*) are supported. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **Create**.

Task	Operation
Deleting a table	1. Enter the policy name in **Policy Name**. 2. Enter and select the corresponding database on the right of **database**. (If you want to delete a database, enter the name of the database to be created or enter ***** to indicate a database with any name, and then select the name.) Enter and select the corresponding table name on the right of **table** and **column**. Wildcard characters (*) are supported. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **Drop**. **NOTE** For CarbonData tables, only the owner of the corresponding database or table can perform the **drop** operation.
ALTER operation	1. Enter the policy name in **Policy Name**. 2. Enter and select the corresponding database on the right of **database**, enter and select the corresponding table on the right of **table**, and enter and select the corresponding column name on the right of **column**. Wildcard characters (*) are supported. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **Alter**.
LOAD operation	1. Enter the policy name in **Policy Name**. 2. Enter and select the corresponding database on the right of **database**, enter and select the corresponding table on the right of **table**, and enter and select the corresponding column name on the right of **column**. Wildcard characters (*) are supported. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **update**.

Task	Operation
INSERT operation	1. Enter the policy name in **Policy Name**. 2. Enter and select the corresponding database on the right of **database**, enter and select the corresponding table on the right of **table**, and enter and select the corresponding column name on the right of **column**. Wildcard characters (*) are supported. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **update**. 5. The user also needs to have the **submit-app** permission of the Yarn task queue. By default, the Hadoop user group has the **submit-app** permission of all Yarn task queues. For details about how to load a network instance to a cloud connection, see **Adding a Ranger Access Permission Policy for Yarn**.
GRANT operation	1. Enter the policy name in **Policy Name**. 2. Enter and select the corresponding database on the right of **database**, enter and select the corresponding table on the right of **table**, and enter and select the corresponding column name on the right of **column**. Wildcard characters (*) are supported. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Select **Delegate Admin**.
ADD JAR operation	1. Enter the policy name in **Policy Name**. 2. Click **database**, and select **global** from the drop-down list. On the right of **global**, enter related information and select ***. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **Temporary UDF Admin**.
VIEW and **INDEX** permissions	1. Enter the policy name in **Policy Name**. 2. On the right side of **database**, enter the database name and select the corresponding database. (If you want to delete a database, enter the database name and select ***.) On the right side of **table**, enter a table name and select the view and index names. On the right side of **column**, enter a Hive column name, and select ***. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select permissions for the user as required.

Task	Operation
Operations on other user database tables	1. Perform the preceding operations to add the corresponding permissions. 2. Grant the read, write, and execution permissions on the HDFS paths of other user database tables to the current user. For details, see **Adding a Ranger Access Permission Policy for HDFS**.

NOTE

After Spark SQL access policy is added on Ranger, you need to add the corresponding path access policies in the HDFS access policy. Otherwise, data files cannot be accessed. For details, see **Adding a Ranger Access Permission Policy for HDFS**.

- The global policy in the Ranger policy is only used to associate with the **Temporary UDF Admin** permission to control the upload of UDF packages.
- When Ranger is used to control Spark SQL permissions, the **empower** syntax is not supported.
- Ranger policies do not support local paths or HDFS paths containing spaces.
- With Ranger authentication activated, default permissions for related tables are required to perform operations on a view. To enable independent authentication for a view, bypassing table permissions, set parameter **spark.ranger.plugin.viewaccesscontrol.enable** to **true**.
  - When submitting jobs in non-Spark-beeline mode, you need to set this parameter in the *Client installation directory*/**Spark/spark/conf/spark-defaults.conf** file.
  - When submitting jobs in Spark-beeline mode, you need to set this parameter in the *Client installation directory*/**Spark/spark/conf/spark-defaults.conf** file. You need to choose **JDBCserver** > **Customization** to add this parameter as well.

Step 5 Click **Add** to view the basic information about the policy in the policy list. After the policy takes effect, check whether the related permissions are normal.

To disable a policy, click  to edit the policy and set the policy to **Disabled**.

If a policy is no longer used, click  to delete it.

----End

Data Masking of the Spark2x Table

Ranger supports data masking for Spark2x data. It can process the returned result of the **select** operation you performed to mask sensitive information.

Step 1 Change the value of **spark.ranger.plugin.masking.enable** to **true** on the server and client, respectively.

- Server: Log in to FusionInsight Manager, choose **Clusters** > **Services** and click the **Spark2x** component. On the displayed page, click the **Configurations** tab and click the **All Configurations** tab. Search for **spark.ranger.plugin.masking.enable** and change the value to **true**. Save the modifications, and restart the service.

- Client: Log in to the Spark client node, go to the *Client installation directory*/**Spark/spark/conf** directory, find the **spark-defaults.conf** file, and change the value of **spark.ranger.plugin.masking.enable** to **true**.

Step 2 Log in to the Ranger WebUI and click the component plug-in name, for example, **Hive**, in the **HADOOP SQL** area on the home page.

Step 3 On the **Masking** tab page, click **Add New Policy** to add a Spark2x permission control policy.

Step 4 Configure the parameters listed in the table below based on the service demands.

Table 23-16 Spark2x data masking parameters

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service.
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**,**192.168.1.20**, or **192.168.1.***.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
Hive Database	Name of the Spark2x database to which the current policy applies.
Hive Table	Name of the Spark2x table to which the current policy applies.
Hive Column	Name of the Spark2x column to which the current policy applies.
Description	Policy description.
Audit Logging	Whether to audit the policy.

Parameter	Description
Mask Conditions	In the **Select Group** and **Select User** columns, select the user group or user to which the permission is to be granted, click **Add Conditions**, add the IP address range to which the policy applies, then click **Add Permissions**, and select **select**. Click **Select Masking Option** and select a data masking policy. • Redact: Use **x** to mask all letters and **0** to mask all digits. • Partial mask: show last 4: Only the last four characters are displayed. • Partial mask: show first 4: Only the first four characters are displayed. • Hash: Perform hash calculation for data. • Nullify: Replace the original value with the NULL value. • Unmasked(retain original value): The original data is displayed. • Date: show only year: Only the year information is displayed. • Custom: You can use any valid Hive UDF (returns the same data type as the data type in the masked column) to customize the policy. To add a multi-column masking policy, click  .
Deny Conditions	Policy rejection condition, which is used to configure the permissions and exceptions to be denied in the policy. The configuration method is similar to that of **Allow Conditions**.

----End

Spark2x Row-Level Data Filtering

Ranger allows you to filter data at the row level when you perform the **select** operation on Spark2x data tables.

Step 1 Change the value of **spark.ranger.plugin.rowfilter.enable** to **true** on the server and client, respectively.

- Server: Log in to FusionInsight Manager, choose **Clusters** > **Services** and click the **Spark2x** component. On the displayed page, click the **Configurations** tab and click the **All Configurations** tab. Search for **spark.ranger.plugin.rowfilter.enable** and change the value to **true**. Save the modifications, and restart the service.
- Client: Log in to the Spark client node, go to the *Client installation directory*/**Spark/spark/conf** directory, find the **spark-defaults.conf** file, and change the value of **spark.ranger.plugin.rowfilter.enable** to **true**.

Step 2 Log in to the Ranger WebUI and click the component plug-in name, for example, **Hive**, in the **HADOOP SQL** area on the home page.

Step 3 On the **Row Level Filter** tab page, click **Add New Policy** to add a row data filtering policy.

Step 4 Configure the parameters listed in the table below based on the service demands.

Table 23-17 Parameters for filtering Spark2x row data

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service.
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**,**192.168.1.20**, or **192.168.1.***.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
Hive Database	Name of the Spark database to which the current policy applies. Note that only one database name can be used and the wildcard character (*) is not allowed.
Hive Table	Name of the Spark table to which the current policy applies. Note that only one table name can be specified and the wildcard character (*) is not permitted.
Description	Policy description.
Audit Logging	Whether to audit the policy.
Row Filter Conditions	In the **Select Role**, **Select Group**, and **Select User** columns, select the object to which the permission is to be granted, click **Add Conditions**, add the IP address range to which the policy applies, then click **Add Permissions**, and select **select**. Click **Row Level Filter** and enter data filtering rules. For example, if you want to filter the data in the **zhangsan** row in the **name** column of **table A**, the filtering rule is **name <>'zhangsan'**. For more information, see the official Ranger document. To add more rules, click  .

Step 5 Click **Add** to view the basic information about the policy in the policy list.

Step 6 After you perform the **select** operation on a table configured with a data masking policy on the Spark2x client, the system processes and displays the data.

----End

23.4.7 Adding a Ranger Access Permission Policy for Kafka

Scenario

Ranger administrators can use Ranger to configure the read, write, and management permissions of the Kafka topic and the management permission of

the cluster for the Kafka user. This section describes how to add the production permission of the **test** topic for the **test** user.

Prerequisites

- The Ranger service has been installed and is running properly.
- You have created users, user groups, or roles for which you want to configure permissions.

Procedure

Step 1 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

Step 2 On the home page, click the component plug-in name in the **KAFKA** area, for example, **Kafka**.

Step 3 Click **Add New Policy** to add a Kafka permission control policy.

Step 4 Configure the following parameters based on the service demands.

Table 23-18 Kafka permission parameters

Parameter	Description
Policy Type	Access type.
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**,**192.168.1.20**, or **192.168.1.***.
Policy Name	Policy name, which can be customized and must be unique in the service.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
topic	Name of the topic applicable to the current policy. You can enter multiple values. The value can contain wildcards, such as **test**, **test***, and *. The **Include** policy applies to the current input object, and the **Exclude** policy applies to objects other than the current input object.
Description	Policy description.
Audit Logging	Whether to audit the policy.

Parameter	Description
Allow Conditions	Permission and exception conditions allowed by a policy. The priority of an exception condition is higher than that of a normal condition. In the **Select Role**, **Select Group**, and **Select User** columns, select the role, user group, or user to which you want to assign permissions. Click **Add Conditions**, add the IP address range to which the policy applies, and click **Add Permissions** to add corresponding permissions. ● Publish: production permission ● Consume: consumption permission ● Describe: query permission ● Create: topic creation permission ● Delete: topic deletion permission ● Describe Configs: configuration query permission ● Alter: permission to change the number of partitions of a topic. ● Alter Configs: configuration modification permission ● Select/Deselect All: Select or deselect all. To add multiple permission control rules, click  . If users or user groups in the current condition need to manage this policy, select **Delegate Admin**. These users will become the agent administrators. The agent administrators can update and delete this policy and create sub-policies based on the original policy.
Deny Conditions	Policy rejection condition, which is used to configure the permissions and exceptions to be denied in the policy. The configuration method is the same as that of **Allow Conditions**. The priority of the rejection condition is higher than that of the allowed conditions configured in **Allow Conditions**.

For example, to add the production permission for the **test** topic of user **testuser**, configure the following information:

Figure 23-4 Kafka permission parameters

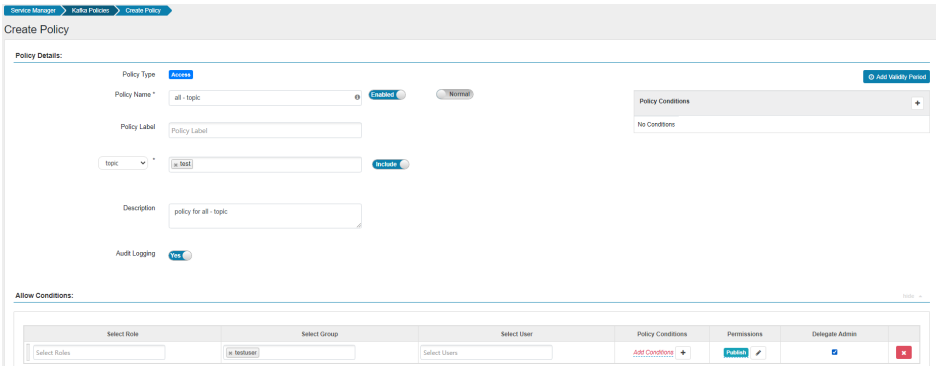


Table 23-19 Setting permissions

Scenario	Role Authorization
Setting the Kafka administrator permissions	1. On the home page, click the component plug-in name in the **KAFKA** area, for example, **Kafka**. 2. Select the policy whose **Policy Name** is **all - topic** and click  to edit the policy. 3. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 4. Click **Add Permissions** and select **Select/ Deselect All**.
Setting the permission for a user to create a topic	1. Specify a topic name in **topic**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Create**. **NOTE** Currently, the Kafka kernel supports the **--zookeeper** and **--bootstrap-server** methods to create topics. The **--zookeeper** method will be deleted from the community in later versions. Therefore, you are advised to use the **--bootstrap-server** method to create topics. Note: Currently, Kafka supports only the authentication of topic creation in **--bootstrap-server** mode and does not support that in **--zookeeper** mode.

Scenario	Role Authorization
Setting the permission for a user to delete a topic	1. Specify a topic name in **topic**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Delete**. **NOTE** Currently, the Kafka kernel supports the **--zookeeper** and **--bootstrap-server** methods to delete topics. The **--zookeeper** method will be deleted from the community in later versions. Therefore, you are advised to use the **--bootstrap-server** method to delete topics. Note: Currently, Kafka supports only the authentication of topic deletion in **--bootstrap-server** mode and does not support that in **--zookeeper** mode.
Setting the permission for a user to query a topic	1. Specify a topic name in **topic**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Describe** and **Describe Configs**. **NOTE** Currently, the Kafka kernel supports the **--zookeeper** and **--bootstrap-server** methods to query topics. The **--zookeeper** method will be deleted from the community in later versions. Therefore, you are advised to use the **--bootstrap-server** method to query topics. Note: Currently, Kafka supports only the authentication of topic query in **--bootstrap-server** mode and does not support that in **--zookeeper** mode.
Setting the production permission of a user on a topic	1. Specify a topic name in **topic**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Publish**.
Setting the consumption permission of a user on a topic	1. Specify a topic name in **topic**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Consume**. **NOTE** During topic consumption, offset management is involved. Therefore, the **Consume** permission of **ConsumerGroup** must be enabled at the same time. For details, see "Setting a User's Permission to Submit ConsumerGroup Offsets".
Setting the permission for a user to expand a topic (by adding partitions)	1. Specify a topic name in **topic**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Alter**.

Scenario	Role Authorization
Setting the permission for a user to modify the topic configuration	Currently, the Kafka kernel does not support to modify topic parameters based on **--bootstrap-server**. Therefore, Ranger does not support authentication for this behavior.
Setting all the management permissions of a user on a cluster	1. Enter a cluster name and select the cluster on the right side of **cluster**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Kafka Admin**.
Setting the permission for a user to create a cluster	1. On the home page, click the component plug-in name in the **KAFKA** area, for example, **Kafka**. 2. Select the policy whose **Policy Name** is **all - cluster** and click  to edit the policy. 3. Enter a cluster name and select the cluster on the right side of **cluster**. 4. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 5. Click **Add Permissions** and select **Create**. **NOTE** The authentication of the **Create** operation of a cluster involves the following two scenarios: 1. After the **auto.create.topics.enable** parameter is enabled in the cluster, the client sends data to a topic that has not been created in the service. In this case, the system checks whether the user has the **Create** permission of the cluster. 2. If a user creates a large number of topics and is granted the **Cluster Create** permission, the user can create any topic in the cluster.
Setting the permission for a user to modify the cluster configuration	1. Enter a cluster name and select the cluster on the right side of **cluster**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Alter Configs**. **NOTE** The configuration modification permission allows you to modify the Broker and Broker Logger configurations. After the configuration modification permission is granted to a user, the user can query configuration details even if the user does not have the query permission. (The configuration modification permission includes the configuration query permission.)

Scenario	Role Authorization
Setting the permission for a user to query the cluster configuration	1. Enter a cluster name and select the cluster on the right side of **cluster**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Describe** and **Describe Configs**. **NOTE** You can only query Broker and Broker Logger information in the cluster, excluding topics.
Setting the Idempotent Write permission in a cluster for a user	1. Enter a cluster name and select the cluster on the right side of **cluster**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Idempotent Write**. **NOTE** This permission authenticates the **Idempotent Produce** behavior of the user's client.
Setting the permission to migrate partitions in a cluster for a user	1. Enter a cluster name and select the cluster on the right side of **cluster**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Alter**. **NOTE** The **Alter** permission of a cluster can be used to control permissions in the following scenarios: 1. In the **Partition Reassign** scenario, migrate the storage directory of replicas. 2. Elect a leader replica in each partition of the cluster. 3. Add or delete ACLs. Operations in scenarios **Step 4.1** and **Step 4.2** are between a controller and broker and between brokers in the cluster. When a cluster is created, this permission is granted to the built-in Kafka user by default. It is meaningless for a common user to be granted with this permission. Scenario **Step 4.3** involves the ACL management. ACLs are designed for authentication. Currently, Kafka authentication is hosted to Ranger. Therefore, this scenario is not involved (the configuration does not take effect).

Scenario	Role Authorization
Setting the Cluster Action permission in a cluster for a user	1. Enter a cluster name and select the cluster on the right side of **cluster**. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Cluster Action**. **NOTE** This permission controls the synchronization between the leader and follower replicas in the cluster and the communication between nodes. It has been granted to the built-in Kafka user during cluster creation. It is meaningless for a common user to grant this permission.
Setting the TransactionalId permission for a user	1. On the home page, click the component plug-in name in the **KAFKA** area, for example, **Kafka**. 2. Select the policy whose **Policy Name** is **all - transactionalid** and click  to edit the policy. 1. Set **transactionalid** to a transaction ID. 2. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 3. Click **Add Permissions** and select **Publish** and **Describe**. **NOTE** The **Publish** permission is used to authenticate client requests for which the transaction feature is enabled, for example, starting and ending a transaction, submitting an offset, and generating transactional data. The **Describe** permission is used to authenticate the requests from the client and coordinator that have enabled the transaction feature. If the transaction feature is enabled, you are advised to grant both the **Publish** and **Describe** permissions to users.

Scenario	Role Authorization
Setting the DelegationToken permission for a user	1. On the home page, click the component plug-in name in the **KAFKA** area, for example, **Kafka**. 2. Select the policy whose **Policy Name** is **all - delegationtoken** and click  to edit the policy. 3. Set **delegationtoken** to a delegation token. 4. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 5. Click **Add Permissions** and select **Describe**. **NOTE** Currently, Ranger only controls the query permission of DelegationToken, but does not control its **create**, **renew**, and **expire** permissions.
Setting the permission for a user to query ConsumerGroup Offsets	1. On the home page, click the component plug-in name in the **KAFKA** area, for example, **Kafka**. 2. Select the policy whose **Policy Name** is **all - consumergroup** and click  to edit the policy. 3. In **consumergroup**, configure the consumer group to be managed. 4. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 5. Click **Add Permissions** and select **Describe**.
Set the user's submission permission on **ConsumerGroup Offsets**.	1. On the home page, click the component plug-in name in the **KAFKA** area, for example, **Kafka**. 2. Select the policy whose **Policy Name** is **all - consumergroup** and click  to edit the policy. 3. In **consumergroup**, configure the consumer group to be managed. 4. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 5. Click **Add Permissions** and select **Consume**. **NOTE** After a user is granted with the **Consume** permission of **ConsumerGroup**, the user is also granted with the **Describe** permission.

Scenario	Role Authorization
Setting the permission for a user to delete ConsumerGroup Offsets	1. On the home page, click the component plug-in name in the **KAFKA** area, for example, **Kafka**. 2. Select the policy whose **Policy Name** is **all - consumergroup** and click  to edit the policy. 3. In **consumergroup**, configure the consumer group to be managed. 4. In the **Allow Conditions** area, select a user from the **Select User** drop-down list. 5. Click **Add Permissions** and select **Delete**. **NOTE** When a user is granted with the **Delete** permission of **ConsumerGroup**, the user is also granted with the **Describe** permission.

Step 5 (Optional) Add the validity period of the policy. Click **Add Validity period** in the upper right corner of the page, set **Start Time** and **End Time**, and select **Time Zone**. Click **Save**. To add multiple policy validity periods, click  . To delete a policy validity period, click  .

Step 6 Click **Add** to view the basic information about the policy in the policy list. After the policy takes effect, check whether the related permissions are normal.

To disable a policy, click  to edit the policy and set the policy to **Disabled**.

If a policy is no longer used, click  to delete it.

----End

23.4.8 Adding a Ranger Access Permission Policy for HetuEngine

Scenario

Ranger administrators can use Ranger to configure the permission to manage databases, tables, and columns of data sources for HetuEngine users.

Prerequisites

- The Ranger service has been installed and is running properly.
- You have created users, user groups, or roles for which you want to configure permissions.
- The users have been added to the **hetuuser** group.
- Before using HetuEngine, ensure that the client operator or user in the configuration file for connecting to the data source has the expected

operation permission. If the user does not have it, configure the permission by referring to the corresponding data source permission requirements.

Procedure

Step 1 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

Step 2 On the homepage, click **HetuEngine** in the **PRESTO** area.

Step 3 On the **Access** tab page, click **Add New Policy** to add a HetuEngine permission control policy.

Step 4 Configure the parameters listed in the table below based on the service demands.

Granting the access policy to the catalog where the table is located is a basic policy and must be configured before you configure other policies. For details, see **Table 23-21**.

Table 23-20 HetuEngine permission parameters

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service. • **Enabled**: Enable the current policy. • **Disabled**: Disable the current policy.
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**,**192.168.1.20**, or **192.168.1.***.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
Presto Catalog	Name of the data source catalog to which the policy applies. If this parameter is set to *, the policy applies to all catalogs. • **Include**: The policy applies to the current input object. • **Exclude**: The policy applies to objects other than the current input.
Schema	Name of the schema to which the policy applies. The value * indicates all schemas. • **Include**: The policy applies to the current input object. • **Exclude**: The policy applies to objects other than the current input.

Parameter	Description
table	Name of the table or view to which the policy applies. If this parameter is set to *, the policy applies to all tables. • **Include**: The policy applies to the current input object. • **Exclude**: The policy applies to objects other than the current input.
Column	Name of the column to which the policy applies. The value * indicates all columns.
Description	Policy description.
Audit Logging	Whether to audit the policy.
Allow Conditions	Policy allowed condition. You can configure permissions and exceptions allowed by the policy. In the **Select Role**, **Select Group**, and **Select User** columns, select the role, user group, or user to which you want to assign permissions. Click **Add Conditions**, add the IP address range to which the policy applies, and click **Add Permissions** to add corresponding permissions. • **Select**: permission to query data • **Insert**: permission to insert data • **Create**: permission to create data • **Drop**: permission to drop data • **Delete**: permission to delete data • **Use**: permission to use data • **Alter**: permission to alter data • **Update**: permission to update data • Admin: the **Admin** permission • **All**: all permissions (including the **Admin** permission) • **Select/Deselect All**: Select or deselect all. To add multiple permission control rules, click  . If users or user groups in the current condition need to manage this policy, select **Delegate Admin**. These users will become the agent administrators. The agent administrators can update and delete this policy and create sub-policies based on the original policy.
Deny Conditions	Policy rejection condition, which is used to configure the permissions and exceptions to be denied in the policy. The configuration method is the same as that of **Allow Conditions**.

Table 23-21 Setting permissions

Task	Role Authorization
Granting the access policy to the catalog where the table is located	1. Enter the policy name in **Policy Name**. 2. In **Presto Catalog**, enter the catalog of the resource to be authorized, for example, **hive**. 3. Enter the authorized Hetu user in the **Select User** text box. 4. In **Permissions**, select **Select**. **NOTE** This policy is a basic policy. Before configuring other policies, ensure that this policy has been configured.
Granting the permission to access the remote HetuEngine table	1. Enter the policy name in **Policy Name**. 2. In **Presto Catalog**, enter the catalog of the table to be authorized, for example, **systemremote** and **svc**. 3. Select **schema** from the drop-down list box under **Presto Catalog** and enter ***** in the text box. 4. Select **table** from the drop-down list box under **schema** and enter ***** in the text box. 5. Select **column** from the drop-down list box under **table** and enter ***** in the text box. 6. Enter the authorized remote HetuEngine user in the **Select User** text box. 7. In **Permissions**, select **Create**, **Drop**, **Select**, and **Insert**. **NOTE** This policy is a basic policy for remote HetuEngine tables. Before configuring other policies, ensure that this policy has been configured.
Create schemas	1. Enter the policy name in **Policy Name**. 2. In **Presto Catalog**, enter the catalog of the target table to be authorized, for example, **hive**. 3. Select **schema** from the drop-down list box under **Presto Catalog** and enter the name of the target schema to be authorized in the text box. If this parameter is set to *****, all schemas under the current catalog are authorized. 4. Enter the authorized Hetu user in the **Select User** text box. 5. In **Permissions**, select **Create**.

Task	Role Authorization
Drop schemas	1. Enter the policy name in **Policy Name**. 2. In **Presto Catalog**, enter the catalog of the target table to be authorized, for example, **hive**. 3. Select **schema** from the drop-down list box under **Presto Catalog** and enter the name of the target schema to be authorized in the text box. If this parameter is set to *****, all schemas under the current catalog are authorized. 4. Enter the authorized Hetu user in the **Select User** text box. 5. In **Permissions**, select **Drop**.
Create table	1. Enter the policy name in **Policy Name**. 2. In **Presto Catalog**, enter the catalog of the target table to be authorized, for example, **hive**. 3. Select **schema** from the drop-down list box under **Presto Catalog** and enter the name of the schema where the target table to be authorized resides in the text box, for example, **default**. 4. Select **table** from the drop-down list box under **schema** and enter the name of the target table to be authorized in the text box. If this parameter is set to *****, all tables under the current schema are authorized. 5. Enter the authorized Hetu user in the **Select User** text box. 6. In **Permissions**, select **Create**.
Drop tables	1. Enter the policy name in **Policy Name**. 2. In **Presto Catalog**, enter the catalog of the target table to be authorized, for example, **hive**. 3. Select **schema** from the drop-down list box under **Presto Catalog** and enter the name of the schema where the target table to be authorized resides in the text box, for example, **default**. 4. Select **table** from the drop-down list box under **schema** and enter the name of the target table to be authorized in the text box. If this parameter is set to *****, all tables under the current schema are authorized. 5. Enter the authorized Hetu user in the **Select User** text box. 6. In **Permissions**, select **Drop**.

Task	Role Authorization
Alter tables	1. Enter the policy name in **Policy Name**. 2. In **Presto Catalog**, enter the catalog of the target table to be authorized, for example, **hive**. 3. Select **schema** from the drop-down list box under **Presto Catalog** and enter the name of the schema where the target table to be authorized resides in the text box, for example, **default**. 4. Select **table** from the drop-down list box under **schema** and enter the name of the target table to be authorized in the text box. If this parameter is set to *****, all tables under the current schema are authorized. 5. Enter the authorized Hetu user in the **Select User** text box. 6. In **Permissions**, select **Alter**. **NOTE** **ALTER TABLE table_name DROP [IF EXISTS] PARTITION partition_spec[, PARTITION partition_spec, ...];** requires the table-level **delete** and column-level **select** permissions.
Show tables	1. Enter the policy name in **Policy Name**. 2. In **Presto Catalog**, enter the catalog of the target table to be authorized, for example, **hive**. 3. Select **schema** from the drop-down list box under **Presto Catalog** and enter the name of the target schema that allows to show table in the text box, for example, **default**. 4. Enter the authorized Hetu user in the **Select User** text box. 5. In **Permissions**, select **Select**.
Insert tables	1. Enter the policy name in **Policy Name**. 2. In **Presto Catalog**, enter the catalog of the target table to be authorized, for example, **hive**. 3. Select **schema** from the drop-down list box under **Presto Catalog** and enter the name of the schema where the target table to be authorized resides in the text box, for example, **default**. 4. Select **table** from the drop-down list box under **schema** and enter the name of the target table to be authorized in the text box. If this parameter is set to *****, all tables under the current schema are authorized. 5. Enter the authorized Hetu user in the **Select User** text box. 6. In **Permissions**, select **Insert**.

Task	Role Authorization
Delete	1. Enter the policy name in **Policy Name**. 2. In **Presto Catalog**, enter the catalog of the target table to be authorized, for example, **hive**. 3. Select **schema** from the drop-down list box under **Presto Catalog** and enter the name of the schema where the target table to be authorized resides in the text box, for example, **default**. 4. Select **table** from the drop-down list box under **schema** and enter the name of the target table to be authorized in the text box. If this parameter is set to *****, all tables under the current schema are authorized. 5. Enter the authorized Hetu user in the **Select User** text box. 6. In **Permissions**, select **Delete**.
Select	1. Enter the policy name in **Policy Name**. 2. In **Presto Catalog**, enter the catalog of the target table to be authorized, for example, **hive**. 3. Select **schema** from the drop-down list box under **Presto Catalog** and enter the name of the schema where the target table to be authorized resides in the text box. 4. Select **table** from the drop-down list box under **schema** and enter the name of the target table to be authorized in the text box. If this parameter is set to *****, all tables under the current schema are authorized. 5. Select **column** from the drop-down list box under **table** and enter the name of the target column to be authorized in the text box. If this parameter is set to *****, all columns under the current table are authorized. 6. Enter the authorized Hetu user in the **Select User** text box. 7. In **Permissions**, select **Select**.

Task	Role Authorization
Show columns	1. Enter the policy name in **Policy Name**. 2. In **Presto Catalog**, enter the catalog of the target table to be authorized, for example, **hive**. 3. Select **schema** from the drop-down list box under **Presto Catalog** and enter the name of the schema where the target table to be authorized resides in the text box, for example, **default**. 4. Select **table** from the drop-down list box under **schema** and enter the name of the target table to be authorized in the text box. If this parameter is set to *****, all tables under the current schema are authorized. 5. Select **column** from the drop-down list box under **table** and enter the name of the target column to be authorized in the text box. If this parameter is set to *****, all columns under the current table are authorized. 6. Enter the authorized Hetu user in the **Select User** text box. 7. In **Permissions**, select **Select**.
Set sessions	1. Enter the policy name in **Policy Name**. 2. Enter ***** in the **Presto Catalog** text box. 3. Enter the authorized Hetu user in the **Select User** text box. 4. Select **Delegate Admin**.

NOTE

- The configuration takes effect about 30 seconds after the permission is configured.
- The current permission control is available to columns.

Step 5 (Optional) Add the validity period of the policy. Click **Add Validity period** in the upper right corner of the page, set **Start Time** and **End Time**, and select **Time Zone**. Click **Save**. To add multiple policy validity periods, click  . To delete a policy validity period, click  .

Step 6 Click **Add** to view the basic information about the policy in the policy list. After the policy takes effect, check whether the related permissions are normal.

To disable a policy, click  to edit the policy and set the policy to **Disabled**.

If a policy is no longer used, click  to delete it.

----End

HetuEngine Data Masking

Ranger supports data masking for HetuEngine data. It can process the return result of the **select** operation performed by a user to mask sensitive information.

Step 1 Log in to the Ranger web UI. Click **HetuEngine** in the **PRESTO** area on the homepage.

Step 2 On the **Masking** tab page, click **Add New Policy** to add a HetuEngine data masking policy.

Step 3 Configure the parameters listed in the table below based on the service demands.

Table 23-22 HetuEngine data masking parameters

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service.
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**,**192.168.1.20**, or **192.168.1.***.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
Presto Catalog	Name of the catalog to which the current policy applies.
Presto Schema	Name of the database to which the current policy applies.
Presto Table	Name of the table to which the current policy applies.
Presto Column	Name of the column to which the current policy applies.
Description	Policy description.
Audit Logging	Whether to audit the policy.

Parameter	Description
Mask Conditions	In the **Select Role**, **Select Group**, and **Select User** columns, select the object to which the permission is to be granted, click **Add Conditions**, add the IP address range to which the policy applies, then click **Add Permissions**, and select **Select**. Click **Select Masking Option** and select a data masking policy. • Redact: Use **x** to mask all letters and **0** to mask all digits. • **Partial mask: show last 4**: Only the last four characters are displayed, and the rest characters are displayed using **x**. • **Partial mask: show first 4**: Only the first four characters are displayed, and the rest characters are displayed using **x**. • **Hash**: Replace the original value with the hash value. • **Nullify**: Replace the original value with the NULL value. • **Unmasked (retain original value)**: Keep the original value. • **Custom**: You customize policies using any valid return data type which is the same as the data type in the masked column. To add a multi-column masking policy, click  .

Step 4 Click **Add** to view the basic information about the policy in the policy list.

Step 5 After a user performs the **select** operation on a table for which a data masking policy has been configured on a HetuEngine client, the system processes the data and displays it.

----End

HetuEngine Row-level Data Filtering

Ranger allows you to filter data at the row level when you perform the select operation on a HetuEngine data table.

Step 1 Log in to the Ranger web UI. Click **HetuEngine** in the **PRESTO** area on the homepage.

Step 2 On the **Row Level Filter** tab page, click **Add New Policy** to add a row data filtering policy.

Step 3 Configure the parameters listed in the table below based on the service demands.

Table 23-23 Parameters for filtering HetuEngine row data

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service.

Parameter	Description
Policy Conditions	IP address filtering policy, which can be customized. You can enter one or more IP addresses or IP address segments. The IP address can contain the wildcard character (*), for example, **192.168.1.10**,**192.168.1.20**, or **192.168.1.***.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
Presto Catalog	Name of the catalog to which the current policy applies.
Presto Schema	Name of the database to which the current policy applies.
Presto Table	Name of the table to which the current policy applies.
Description	Policy description.
Audit Logging	Whether to audit the policy.
Row Filter Conditions	In the **Select Role**, **Select Group**, and **Select User** columns, select the object to which the permission is to be granted, click **Add Conditions**, add the IP address range to which the policy applies, then click **Add Permissions**, and select **Select**. Click **Row Level Filter** and enter data filtering rules. For example, if you want to filter the data in the **zhangsan** row in the **name** column of **table A**, the filtering rule is **name <>'zhangsan'**. For more information, see the official Ranger document. To add more rules, click  .

Step 4 Click **Add** to view the basic information about the policy in the policy list.

Step 5 After a user performs the **select** operation on a table for which a data masking policy has been configured on a HetuEngine client, the system processes the data and displays it.

----End

23.4.9 Adding a Ranger Access Permission Policy for OBS

Scenario

Ranger administrators can use Ranger to configure the read and write permissions on OBS directories or files for OBS users.

NOTE

This section applies only to MRS 3.3.0-LTS or later.

Prerequisites

- The Ranger service has been installed and is running properly.

- You have created a user group for which you want to configure permissions.
- The Guardian service has been installed.

Procedure

Step 1 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

Step 2 On the home page, click the component plug-in name in the **EXTERNAL AUTHORIZATION** area, for example, **OBS**.

Step 3 Click **Add New Policy** to add an OBS permission control policy.

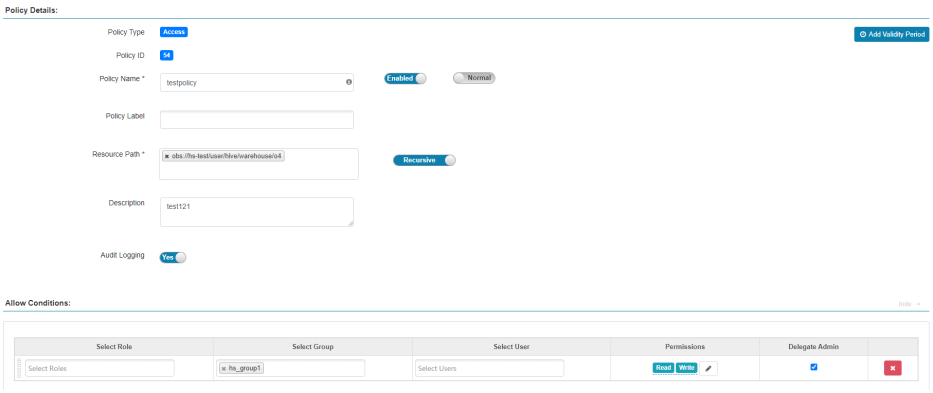
Step 4 Configure the parameters listed in the table below based on service requirements.

Table 23-24 OBS permission parameters

Parameter	Description
Policy Name	Policy name, which can be customized and must be unique in the service.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.
Resource Path	Resource path, which is the OBS path folder to which the current policy applies. You can enter multiple values but cannot use wildcards (*). The configured OBS path folder must exist. Otherwise, the authorization fails. By default, permission recursion is enabled on OBS and cannot be modified. Subdirectories without any permission inherit all permissions of their parent directories.
Description	Policy description.
Audit Logging	Whether to audit the policy.
Allow Conditions	Policy allowed condition. You can configure permissions allowed by the policy. In the **Select Group** column, select the created user group to which you want to grant permissions. (The configuration of **Select Role** or **Select User** does not take effect.) Click **Add Permissions** to add permissions. • **Read**: permission to read data • **Write**: permission to write data • **Select/Deselect All**: permission to select or deselect all To add multiple permission control rules, click  . To delete a permission control rule, click  .

To give user group **hs_group1** the read and write access to the **obs://hs-test/user/hive/warehouse/o4** table, follow the following configuration steps. Note

that user group names can only contain up to 52 characters, including numbers (0 to 9), letters (A to Z or a to z), underscores (_), and number signs (#). Otherwise, the policy will fail to add.



Step 5 Click **Add** to view the basic information about the policy in the policy list. After the policy takes effect, check whether the related permissions are normal.

If a policy is no longer used, click  to delete it.

----End

23.4.10 Hive Tables Supporting Cascading Authorization

This topic is available for MRS 3.3.0 or later versions only. Before using this function, ensure that the following conditions are met:

- The AccessLabel function must be enabled on OBS. For details about how to enable this function, contact OBS O&M personnel.
- To ensure successful OBS authorization for the table, the following conditions must be met when using OBS as the storage source:
  - The Guardian service must have been installed in the cluster.
  - Tables stored in OBS can only be authorized to user groups.
  - OBS cascading authorization can be used in clusters with Kerberos authentication enabled only.

Scenario

Enabling cascading authorization for a cluster greatly enhances authentication usability. You only need to authorize service tables once on the Ranger page, and the system will automatically associate permissions with the data storage source in a fine-grained manner, without the need to detect the storage path of the tables or require secondary authorization. With Ranger, you can authorize and authenticate tables that have separated storage and compute, effectively eliminating the drawbacks of this method. The cascading authorization function of Hive tables is as follows:

- After Ranger cascading authorization is enabled, when creating a policy in Ranger to authorize for a table, you only need to create a Hive policy for the table and do not need to perform secondary authorization on the table's storage source.

- When the storage source of an authorized database or table changes, the database or table is periodically associated with the new storage source (HDFS or OBS) to generate corresponding permissions.

  NOTE

  - Cascading authorization is not supported for view tables.
  - Cascading authorization can be performed only on databases and tables, and cannot be on partitions. If a partition path is not in the table path, you need to manually authorize the partition path.
  - Cascading authorization for **Deny Conditions** in the Hive Ranger policy is not supported. That is, the Deny Conditions permission only restricts the table permission and cannot generate the permission of the HDFS/OBS storage source.
  - A policy whose **database** is ***** and **table** is ***** cannot be created in Hive Ranger.
  - The permission of the HDFS Ranger policy is prior to that of the HDFS/OBS storage source generated by cascading authorization. If the HDFS Ranger permission has been set for the HDFS storage source of the table, the cascading permission does not take effect.
  - If you have cascading authorization on an OBS storage source table, you won't be able to perform the ALTER operation. To use this operation, you must grant **Read** and **Write** permissions to the corresponding user group on the parent directory of the OBS table path. User group names can have up to 52 characters, including numbers (0 to 9), letters (A to Z or a to z), underscores (_), and number signs (#). Otherwise, the policy will fail to add. For how to modify the user group information, see **Creating a User Group**.

Enabling Cascading Authorization

Step 1 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Ranger**, and click **Configurations**.

Step 2 Search for the **ranger.ext.authorization.cascade.enable** parameter and set it to **true**.

Step 3 Click **Save**.

Step 4 Click **Instance** and select all RangerAdmin instances. Click **More** and select **Restart Instance**. Enter the password, and click **OK** to restart all RangerAdmin instances.

----End

Connecting to the HDFS Storage Source

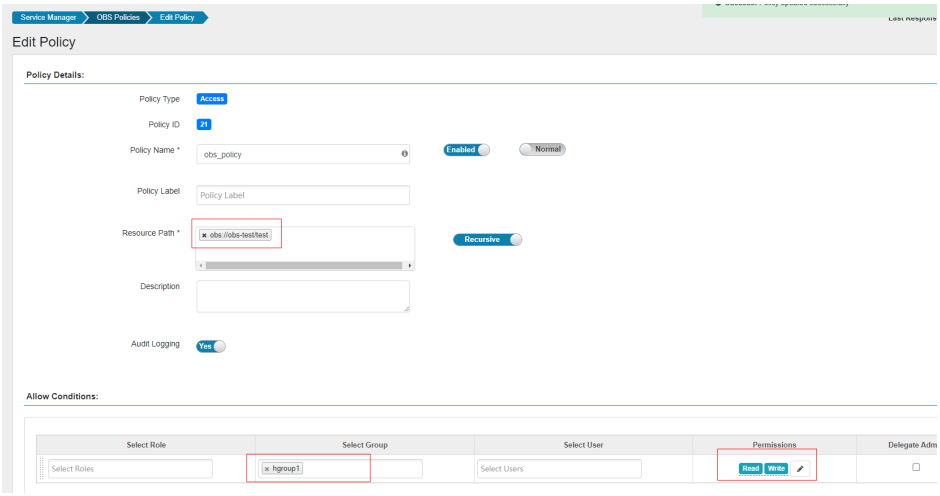
The HDFS storage source does not need to be configured.

Connecting to the OBS Storage Source

- Setting the location to an OBS path when creating a table
  a. Ensure that the storage and compute decoupling has been configured. For details, see "Interconnecting with OBS Using the Guardian Service".
  b. Log in to the Ranger management page as the Ranger administrator **rangeradmin**. On the home page, click **OBS** in the **EXTERNAL AUTHORIZATION** area, click **Add New Policy**, and assign the **Read** and **Write** permissions on the OBS storage path to the user group to which

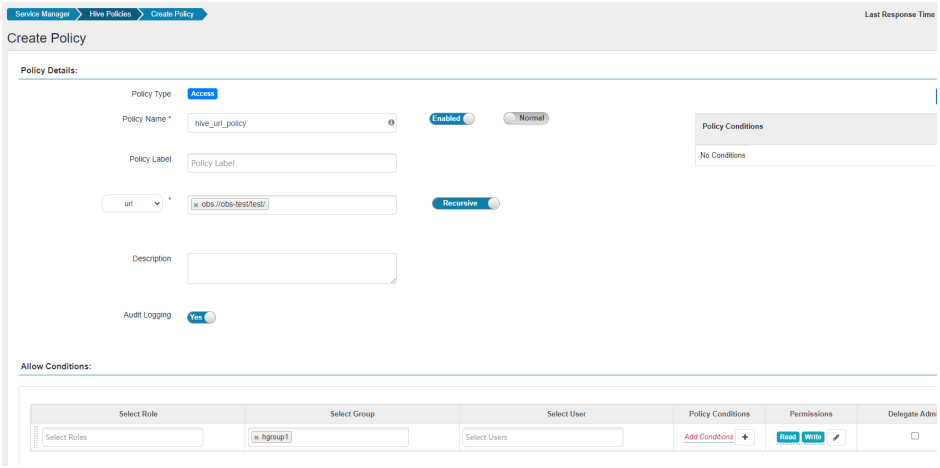
the corresponding user belongs. For details, see **Adding a Ranger Access Permission Policy for OBS**.

For example, assign the **Read** and **Write** permissions on the **obs://obs-test/test/** directory to the **hgroup1** user group, as shown in the following figure.



c. On the home page, click the component plug-in name **Hive** in the **HADOOP SQL** area. On the **Access** page, click **Add New Policy** to add a URL policy that assigns the **Read** and **Write** permissions on OBS storage paths to the user group to which the corresponding user belongs. For details, see **Adding a Ranger Access Permission Policy for Hive**.

For example, create the **hive_url_policy URL** policy for the **hgroup1** user group and assign the **Read** and **Write** permissions on the **obs://obs-test/test/** directory to the user group, as shown in the following figure.



d. Log in to the beeline client and set **Location** to the OBS file system path when creating a table.

cd *Client installation directory*

kinit *Component operation user*

beeline

For example, to create a table named **test** whose **Location** is **obs://obs-test/test/***Database name*/*Table name*, run the following command:

create table test(name string) location "obs://obs-test/test/*Database name***/***Table name***";**

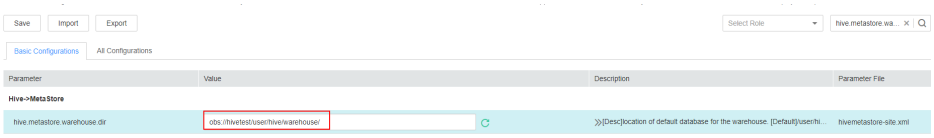
- Interconnecting Hive with OBS through Metastore

  a. Ensure that the storage and compute decoupling has been configured. For details, see "Interconnecting with OBS Using the Guardian Service".

  b. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Hive**, and click **Configurations**.

  c. Search for **hive.metastore.warehouse.dir** in the search box and change the parameter value to an OBS path, for example, **obs://hivetest/user/hive/warehouse/**. **hivetest** indicates the OBS file system name.

  Figure 23-5 hive.metastore.warehouse.dir configuration

  d. Save the configuration, choose **Cluster** > **Services**, and restart the Hive service in the service list.

  e. Update the client configuration file.

     i. Log in to the node where the Hive client is located and run the following command to modify **hivemetastore-site.xml** in the Hive client configuration file directory:

     vi *Client installation directory***/Hive/config/hivemetastore-site.xml**

     ii. Change the value of **hive.metastore.warehouse.dir** to the corresponding OBS path, for example, **obs://hivetest/user/hive/warehouse/**.

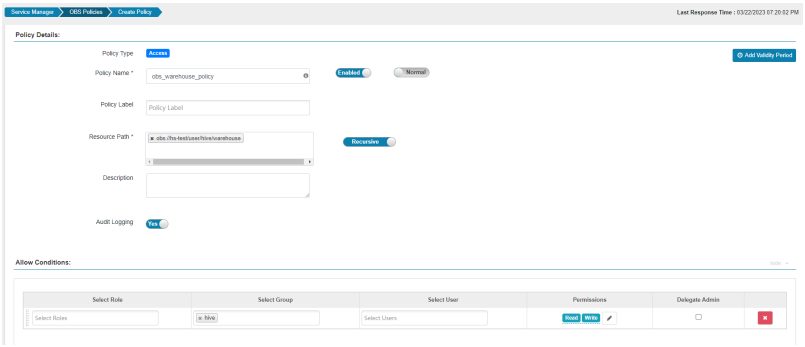
     ```
     </property>
     <property>
     <name>hive.metastore.warehouse.dir</name>
     <value>obs://hivetest/user/hive/warehouse/</value>
     </property>
     <property>
     ```

     iii. Change the value of **hive.metastore.warehouse.dir** of **hivemetastore-site.xml** in the HCatalog client configuration file directory to the corresponding OBS path, for example, **obs://hivetest/user/hive/warehouse/**.

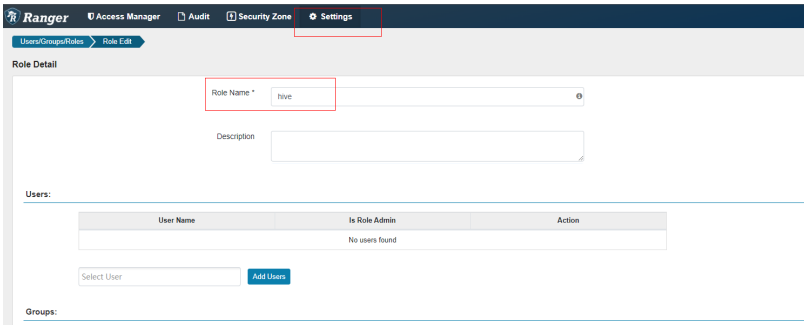
     vi *Client installation directory***/Hive/HCatalog/conf/hivemetastore-site.xml**

     iv. Log in to the Ranger management page as the Ranger administrator **rangeradmin**. On the home page, click **OBS** in the **EXTERNAL AUTHORIZATION** area, click **Add New Policy**, and assign the **Read** and **Write** permissions on the OBS storage path to the user group to which the corresponding user belongs.

     For example, assign the **Read** and **Write** permissions on the **obs://hivetest/user/hive/warehouse/** directory to the **hgroup1** user group:



v. Choose **Settings** > **Roles**, click **Add New Role**, and create a role whose **Role Name** is **hive**.



f. Go to the Hive Beeline CLI, create a table, and ensure that the location is an OBS path.

cd *Client installation directory*

kinit *Component operation user*

beeline

create table test(name string);

desc formatted test;

NOTE

If the current database is located in HDFS, any tables created within it will automatically be located in HDFS without the need to specify the location. To change the default policy for creating tables, update the database location to point to OBS.

The procedure is as follows:

1. Query the location of the database.

   show create database *obs_test*;

   ```
   +----------------------------------------------------+
   |                    createdb_stmt                   |
   +----------------------------------------------------+
   | CREATE DATABASE `obs_test`                         |
   | LOCATION                                           |
   |   'hdfs://hacluster/user/hive/warehouse/obs_test.db'  |
   +----------------------------------------------------+
   3 rows selected (0.038 seconds)
   ```

2. Modify the database location.

   alter database *obs_test* **set location** *'obs://test1/'*

   Run the **show create database** *obs_test* command to check whether the location of the database points to OBS.

   ```
   INFO  : Concurrency mode is disabled, not creating
   +----------------------------+
   |        createdb_stmt       |
   +----------------------------+
   | CREATE DATABASE `obs_test` |
   | LOCATION                   |
   |   'obs://test1/'           |
   +----------------------------+
   3 rows selected (0.063 seconds)
   ```

3. Modify the table location.

   alter table *user_info* **set location** *'obs://test1/'*

   If the table contains data, migrate the original data file to the new location.

23.5 Viewing Ranger Audit Information

Ranger administrators can view audit logs about Ranger running and permission control audit logs after Ranger is used by components for authentication.

Viewing Ranger Audit Information

Step 1 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

Step 2 Choose **Audit** to view the audit information. For details about each tab page, see **Table 23-25**. If there are a large number of audit records, you can filter them in the search box by keyword.

Table 23-25 Audit information

Tab	Description
Access	Currently, MRS does not support online query of audit logs of component resources. You can log in to the component installation node and access **/var/log/Bigdata/audit** to view audit logs of each component.
Admin	Audit information about operations on Ranger, such as creating, updating, and deleting security access policies, creating and deleting component permission policies, and creating, updating, and deleting roles.
Login Sessions	Session audit information about users who log in to Ranger.
Plugins	Component permission policy information in Ranger.
Plugin Status	Audit information about synchronization of the permission policy of each component node.
User Sync	Audit information about synchronization between Ranger and LDAP users.

----End

23.6 Configuring Ranger Security Zone

Security zone can be configured using Ranger. Ranger administrators can divide resources of each component into multiple security zones where Ranger administrators set security policies for specified resources in the zones to facilitate management. Policies defined in a security zone apply only to resources in the zone. After service resources are allocated to the security zone, the access permission policies for the resources in the non-security zone do not take effect. The administrator of a security zone can set policies only in the security zone that the administrator belongs to.

Adding a Security Zone

Step 1 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

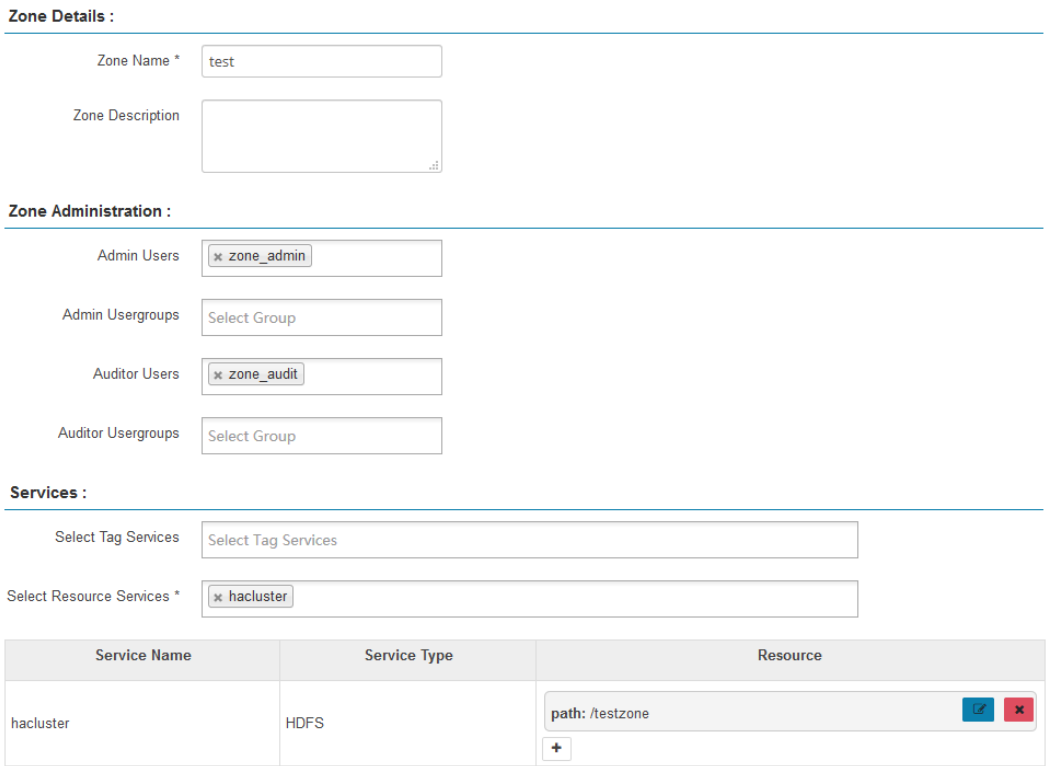
Step 2 Click **Security Zone**. On the zone list page, click  to add a zone.

Table 23-26 Parameters for configuring a security zone

Parameter	Description	Example Value
Zone Name	Security zone	test

Parameter	Description	Example Value
Zone Description	Description of the security zone	-
Admin Users/ Admin Usergroups	Management users and user groups in a security zone. You can add and modify permission policies for related resources in the security zone. At least one user or user group must be configured.	zone_admin
Auditor Users/ Auditor Usergroups	Audit users or user groups to be added. You can view the resource permission policies in the security zone. At least one user or user group must be configured.	zone_user
Select Tag Services	Tag information of a service	-
Select Resource Services	Services and resources in a security zone. After selecting a service, you need to add specific resource objects in the **Resource** column, such as the file directories of the HDFS server, Yarn queues, Hive databases and tables, and HBase tables and columns.	/testzone

For example, to create a security zone for the **/testzone** directory in HDFS, the configuration is as follows:



Step 3 Click **Save** and wait until the security zone is added successfully.

The Ranger administrator can view all security zones on the **Security Zone** page and click **Edit** to modify the attributes of a security zone. If resources do not need to be managed in a security zone, the Ranger administrator can click **Delete** to delete the security zone.

----End

Configuring Permission Policies in a Security Zone

Step 1 Log in to the Ranger management page as the Ranger administrator of a security zone.

Step 2 Select a security zone from the **Security Zone** drop-down list in the upper right corner of the Ranger home page to switch to the permission view of the security zone.

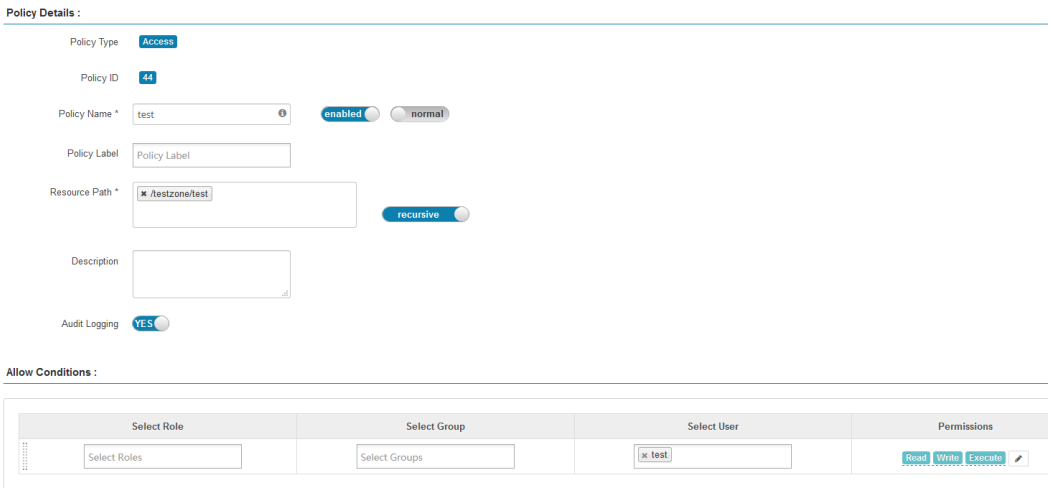
Step 3 Click the permission plug-in name of a component. The page for security access policy list of the component is displayed.

NOTE

In the policy list of each component, the default items generated by the system are automatically inherited to the security zone to ensure the permissions of some default users or user groups in the cluster.

Step 4 Click **Add New Policy** and configure resource access policies for related users or user groups based on the service scenario plan.

In this example, a policy that allows user test to access the **/testzone/test** directory is configured in the security zone.



The following access policies are examples for different components:

- **Adding a Ranger Access Permission Policy for CDL**
- **Adding a Ranger Access Permission Policy for HDFS**
- **Adding a Ranger Access Permission Policy for HBase**
- **Adding a Ranger Access Permission Policy for Hive**
- **Adding a Ranger Access Permission Policy for Yarn**
- **Adding a Ranger Access Permission Policy for Spark2x**
- **Adding a Ranger Access Permission Policy for Kafka**
- **Adding a Ranger Access Permission Policy for HetuEngine**

After the policies are added, wait for about 30 seconds for them to take effect.

NOTE

- Policies defined in a security zone apply only to resources in the zone. After service resources are allocated to the security zone, the access permission policies for the resources in the non-security zone do not take effect.
- To configure access policies for resources outside the current security zone, click **Security Zone** in the upper right corner of the Ranger homepage to exit the current security zone.

----End

23.7 Viewing Ranger User Permission Synchronization Information

You can view Ranger permission settings, such as users, user groups, and roles.

Viewing Ranger Permission Information

Step 1 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**.

Step 2 Choose **Settings** > **Users**/**Groups**/**Roles** to view information about users, user groups, or roles in the system.

- Users: displays all user information synchronized from LDAP or OS to Ranger.
- Groups: displays information about all user groups and role information synchronized from LDAP or OS to Ranger.
- Roles: displays information about roles created in Ranger.

NOTE

- The users, roles, user groups created on FusionInsight Manager are automatically synchronized to Ranger periodically. The default period is 300,000 milliseconds (5 minutes). After roles and user groups in FusionInsight Manager are synchronized to Ranger, they become user groups. Only roles and user groups that are associated with users can be automatically synchronized to Ranger.
- The role created on the Ranger page is a set of users or user groups, which is used to flexibly set the permission access policies of components. The role is different from that on FusionInsight Manager. To prevent any delay in accessing Ranger permission information, it is best to restrict the number of users or user groups assigned to a role to no more than 1,000.
- Users, roles, and user groups cannot be deleted on the Ranger web UI to prevent data inconsistency.

----End

Adjusting Ranger User Types

Step 1 Log in to the Ranger management page.

To change the Ranger user type, you must log in as an **admin** user. For details about the user types, see **Ranger User Type**.

Step 2 Choose **Settings** > **Users**/**Groups**/**Roles**. In the list of users, click the name of the user whose type you want to change.

Step 3 Set **Select Role** to the type to be modified.

Step 4 Click **Save**.

----End

Creating a Ranger Role

Ranger administrators can flexibly configure permission access policies for components based on users, user groups, or roles. User and user group information is automatically synchronized from LDAP, and roles can be manually added.

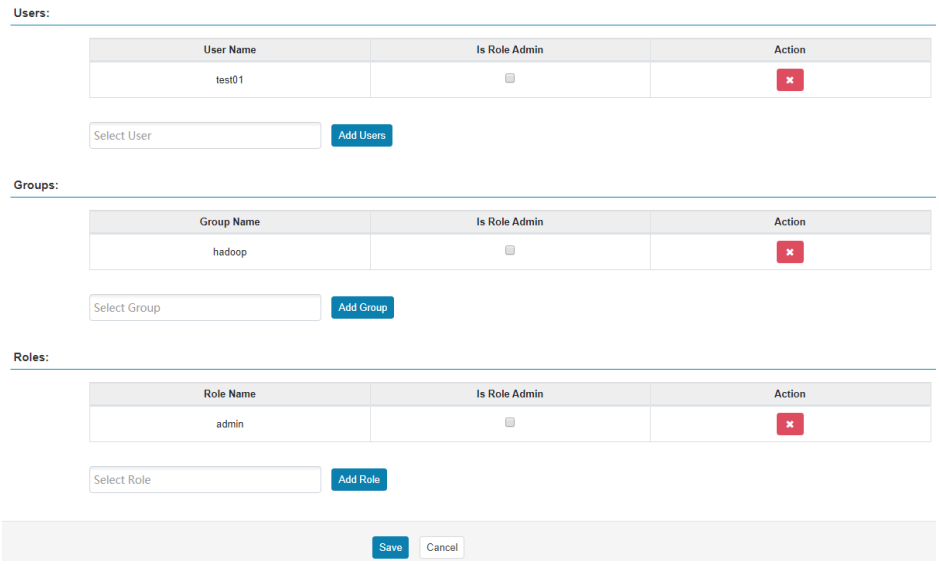
Step 1 Log in to the Ranger management page.

Step 2 Choose **Settings** > **Users**/**Groups**/**Roles** > **Roles** > **Add New Role**.

Step 3 Enter the role name and description as prompted.

Step 4 Add users, user groups, and sub-roles to the role.

- In the **Users** area, select a created user in the system and click **Add Users**.
- In the **Groups** area, select a created user group and click **Add Group**.
- In the **Roles** area, select a created role in the system and click **Add Role**.



Step 5 Click **Save**. The role is added.

NOTE

Added roles cannot be deleted but can be modified.

----End

23.8 Ranger Performance Tuning

Scenario

Ranger provides permission policies for services. When the number of service instances using Ranger increases, you need to adjust the specifications of Ranger.

NOTE

This section applies only to MRS 3.2.0 or later.

Configuring Memory Parameters

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Ranger**. Click **Configurations** then **All Configurations**, and search for **GC_OPTS** in the **RangerAdmin JVM** parameter. The default value of **GC_OPTS** is **-Dproc_rangeradmin -Xms2G -Xmx2G -XX:MaxDirectMemorySize=512M -XX:MetaspaceSize=100M -XX:MaxMetaspaceSize=200M -XX:PermSize=64M -XX:MaxPermSize=512M -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:${RANGER_ADMIN_LOG_DIR}/gc-worker-%p-%t.log -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=20M -verbose:gc -Djdk.tls.ephemeralDHKeySize=3072 -Djava.security.auth.login.config=#{conf_dir}/jaas.conf -Djava.security.krb5.conf=${KRB5_CONFIG} -Dbeetle.application.home.path=${BIGDATA_HOME}/common/runtime/security/config -Djna.tmpdir=${RANGER_TMP_HOME} -Djava.io.tmpdir=${RANGER_TMP_HOME} ${JAVA_STACK_PREFER} -Djdk.tls.rejectClientInitiatedRenegotiation=true**.

Step 2 Change the value of **GC_OPTS** in the **RangerAdmin JVM** parameter as follows:

Service instances that use Ranger include HDFS (NameNode), YARN (ResourceManager), HBase (HMaster and RegionServer), Hive (HiveServer), Kafka (Broker), Elasticsearch (EsNode, EsMaster, and EsClient), CDL (CDLService), and HetuEngine (HSBroker). When the number of these instances increases, change the default value **-Xms2G -Xmx2G** according to the reference RangerAdmin memory specifications listed below.

Ranger Instances	Reference Value
200	-Xms4G -Xmx4G
400	-Xms8G -Xmx8G
600	-Xms12G -Xmx12G

----End

23.9 Ranger Log Overview

Log Description

Log path: The default storage path of Ranger logs is **/var/log/Bigdata/ranger/***Role name*.

- RangerAdmin: /var/log/Bigdata/ranger/rangeradmin (run logs); /var/log/Bigdata/audit/ranger/rangeradmin (audit logs in MRS 3.3.0 and later versions)
- TagSync: /var/log/Bigdata/ranger/tagsync (run logs)
- UserSync: /var/log/Bigdata/ranger/usersync (run logs)
- PolicySync: /var/log/Bigdata/ranger/policysync (run logs in MRS 3.3.0 and later versions)

Log archive rule: The automatic compression and archive function is enabled for Ranger logs. By default, when the size of a log file exceeds 20 MB, the log file is automatically compressed. The naming rule of the compressed log file is as follows: *<Original log file name>-<yyyy-mm-dd_hh-mm-ss>.[ID]***.log.zip**. A maximum of 20 compressed file are retained.

Table 23-27 Ranger log list

Type	Name	Description
RangerAdmin run log file	access_log.*<DATE>*.log	Tomcat access log
	catalina.out	Tomcat service run log
	gc-worker.log	RangerAdmin garbage collection (GC) log

Type	Name	Description
	postinstallDetail.log	Work log generated after an instance is started before installation
	prestartDetail.log	Log that records preparations before instance startup
	ranger-admin-*<hostname>*.log	RangerAdmin run log
	ranger_admin_sql-*<hostname>*.log	RangerAdmin log used to retrieve DBService
	startDetail.log	Instance startup log
TagSync run log	cleanupDetail.log	Instance clearing log
	gc-worker.log	GC log file of an instance
	postinstallDetail.log	Work log generated after an instance is started before installation
	prestartDetail.log	Log that records preparations before instance startup
	ranger-tagsync-*<hostname>*.log	TagSync run log
	startDetail.log	Instance startup log
	tagsync.out	TagSync run log
UserSync run log	auth.log	UnixAuth service run log
	cleanupDetail.log	Instance clearing log
	gc-worker.log	GC log file of an instance
	postinstallDetail.log	Work log generated after an instance is started before installation
	prestartDetail.log	Log that records preparations before instance startup
	ranger-usersync-*<hostname>*.log	UserSync run log
	startDetail.log	Instance startup log

Type	Name	Description
PolicySync run logs (MRS 3.3.0 and later versions)	cleanupDetail.log	Instance clearing log
	policysync.out	Instance run log
	postinstallDetail.log	Work log generated after an instance is started before installation
	prestartDetail.log	Log that records preparations before instance startup
	ranger-policysync.log	Instance run log
	startDetail.log	Instance startup log
	gc-worker-pid*<PID>*-*<Date>*.log.*<ID>*	GC log file of an instance
	stopDetail.log	Instance stopping log
Audit logs (MRS 3.3.0 and later versions)	rangeradmin-audit.log	RangerAdmin audit log

Log Levels

Table 23-28 describes the log levels provided by HDFS. The priorities of log levels are FATAL, ERROR, WARN, INFO, and DEBUG in descending order. Logs whose levels are higher than or equal to the specified level are printed. The number of printed logs decreases as the specified log level increases.

Table 23-28 Log levels

Level	Description
FATAL	Logs of this level record fatal error information about the current event processing that may result in a system crash.
ERROR	Logs of this level record error information about the current event processing, which indicates that system running is abnormal.
WARN	Logs of this level record abnormal information about the current event processing. These abnormalities will not result in system faults.
INFO	Logs of this level record normal running status information about the system and events.

Level	Description
DEBUG	Logs of this level record the system information and system debugging information.

To modify log levels, perform the following operations:

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **Ranger** > **Configurations**.

Step 3 Select **All Configurations**.

Step 4 On the menu bar on the left, select the log menu of the target role.

Step 5 Select a desired log level.

Step 6 Click **Save**. In the displayed dialog box, click **OK** to make the configuration take effect.

NOTE

The configurations take effect immediately without the need to restart the service.

----End

Log Formats

The following table lists the Ranger log formats.

Table 23-29 Log formats

Type	Format	Example Value
Run log	*<yyyy-MM-dd HH:mm:ss,SSS>*|*<Log level>*|*<Name of the thread that generates the log>*|*<Message in the log>*|*<Location where the log event occurs>*	2020-04-29 20:09:28,543 | INFO | http-bio-21401-exec-56 | Request comes from API call, skip cas filter. | CasAuthenticationFilter-Wrapper.java:25

23.10 Common Issues About Ranger

23.10.1 How Do I Determine Whether the Ranger Authentication Is Used for a Service?

Question

How do I determine whether the Ranger authentication is enabled for a service that supports the authentication?

Answer

Log in to FusionInsight Manager and choose **Cluster** > **Services** > *Name of the desired service*. On the service details page, click **More** and check whether the **Enable Ranger** option is available.

- If yes, the Ranger authentication plug-in is not enabled for the service. You can click **Enable Ranger** to enable the function.
- If no, the Ranger authentication plug-in has been enabled for the service. You can configure the permission policy for accessing the service resources on the Ranger management page.

NOTE

If this option does not exist, the current service does not support the Ranger authentication plug-in and Ranger authentication is disabled.

23.10.2 Why Cannot a New User Log In to Ranger After Changing the Password?

Question

When a new user logs in to Ranger, why is the 401 error reported after the password is changed?

Answer

The UserSync synchronizes user data at an interval of 5 minutes by default. Therefore, a new user created on Manager cannot log in to the Ranger before the user data is successfully synchronized because the Ranger database does not have the user information. The user can log in to the Ranger only after the specified interval ends.

In non-security mode, the Ranger does not synchronize user data from Manager. Therefore, only the **admin** user can log in to the Ranger page.

23.10.3 What Should I Do If I Cannot View the Created MRS User on the Ranger Management Page?

Symptom

An account has been created on MRS Manager. Why can't the account be viewed after I access the Ranger management page?

Answer

The user who logs in to the Ranger management page does not have the permission to view the account. Switch to the **rangeradmin** user or another user who has the Ranger administrator permission.

1. On the Ranger web UI, click the username in the upper right corner and choose **Log Out** to log out of the current user.

2. Log in to the system as user **rangeradmin** (default password: **Rangeradmin@123**) or another user who has the Ranger administrator permissions. For details about the usernames and default passwords, see User User Account List.

23.10.4 What Should I Do If MRS Users Failed to Be Synchronized to the Ranger Web UI

Symptom

The user created on MRS Manager cannot be viewed on the Ranger web UI and can be viewed only after UserSync is restarted.

Answer

The default GC memory of the UserSync process is **-Xms1G -Xmx1G**. Change the value based on the site requirements.

Log in to MRS Manager, choose **Cluster** > **Services** > **Ranger**, click **Configurations**, and click **All Configurations** > **UserSync(Role)** > **System**, and change the value of **GC_OPTS**. For example, change the memory size to **-Xms2G -Xmx2G**.

23.11 Ranger Troubleshooting

23.11.1 Ranger Fails to Be Started During Cluster Installation

Question

Ranger fails to start during cluster installation and the Manager process task list shows database errors like "ERROR: cannot drop sequence X_POLICY_REF_ACCESS_TYPE_SEQ". Measures should be taken to fix the issue and complete the Ranger installation.

Answer

This issue arises when there are two instances of RangerAdmin installed.

If the installation fails, you can log in to Manager, manually restart a RangerAdmin instance, and then restart other instance

23.11.2 Existing HBase Tables Cannot Be Searched Using Wildcards When HBase Permission Policies Are Configured

Question

When a Ranger access permission policy is added for HBase and wildcard characters are used to search for an existing HBase table in the policy, the table cannot be found. The following error is reported in **/var/log/Bigdata/ranger/rangeradmin/ranger-admin-*log**:

```
Caused by: javax.security.sasl.SaslException: No common protection layer between client and server
at com.sun.security.sasl.gsskerb.GssKrb5Client.doFinalHandshake(GssKrb5Client.java:253)
at com.sun.security.sasl.gsskerb.GssKrb5Client.evaluateChallenge(GssKrb5Client.java:186)
at
org.apache.hadoop.hbase.security.AbstractHBaseSaslRpcClient.evaluateChallenge(AbstractHBaseSaslRpcClien
t.java:142)
at org.apache.hadoop.hbase.security.NettyHBaseSaslRpcClientHandler
$2.run(NettyHBaseSaslRpcClientHandler.java:142)
at org.apache.hadoop.hbase.security.NettyHBaseSaslRpcClientHandler
$2.run(NettyHBaseSaslRpcClientHandler.java:138)
at java.security.AccessController.doPrivileged(Native Method)
at javax.security.auth.Subject.doAs(Subject.java:422)
at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1761)
at
org.apache.hadoop.hbase.security.NettyHBaseSaslRpcClientHandler.channelRead0(NettyHBaseSaslRpcClientH
andler.java:138)
at
org.apache.hadoop.hbase.security.NettyHBaseSaslRpcClientHandler.channelRead0(NettyHBaseSaslRpcClientH
andler.java:42)
at
org.apache.hbase.thirdparty.io.netty.channel.SimpleChannelInboundHandler.channelRead(SimpleChannelInbo
undHandler.java:105)
at
org.apache.hbase.thirdparty.io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractCh
annelHandlerContext.java:362)
```

Answer

The value of **hbase.rpc.protection** of the HBase service plug-in on Ranger must be the same as that of **hbase.rpc.protection** on the HBase server.

Step 1 Log in to the Ranger management page. For details, see **Logging In to the Ranger Web UI**.

Step 2 In the **HBASE** area on the home page, click the component plug-in name, for example, the  button of HBase.

Step 3 Search for the configuration item **hbase.rpc.protection** and change its value to the value of **hbase.rpc.protection** on the HBase server.

Step 4 Click **Save**.

----End

24 Using Spark/Spark2x

24.1 Spark Usage Instruction

In MRS 3.3.0-LTS and later versions, the Spark2x component is renamed Spark, and the role names of this component are also changed. For example, JobHistory2x is changed to JobHistory.

Refer to the descriptions and operations related to the component name and role names in the document based on your MRS version.

Spark is an open-source and parallel data processing framework that helps users quickly and easily develop big data applications and perform offline processing, streaming processing, and interactive analysis on data.

Spark has a significant performance advantage over Hadoop.

24.2 Spark User Permission Management

24.2.1 Introduction to SparkSQL User Permissions

SparkSQL Permissions

Similar to Hive, Spark SQL is a data warehouse framework built on Hadoop, providing storage of structured data like structured query language (SQL).

MRS supports users, user groups, and roles. Permission must be assigned to roles and then roles are bound to users or user groups. Users can obtain permissions only by binding a role or joining a group that is bound with a role.

NOTE

- If the current component uses Ranger for permission control, you need to configure permission management policies based on Ranger. For details, see **Adding a Ranger Access Permission Policy for Spark2x**.
- After Ranger authentication is enabled or disabled on Spark2x, you need to restart Spark2x and download the client again or update the client configuration file **spark/conf/spark-defaults.conf**.

  Enable Ranger authentication: **spark.ranger.plugin.authorization.enable=true**

  Disable Ranger authentication: **spark.ranger.plugin.authorization.enable=false**

Permission Management

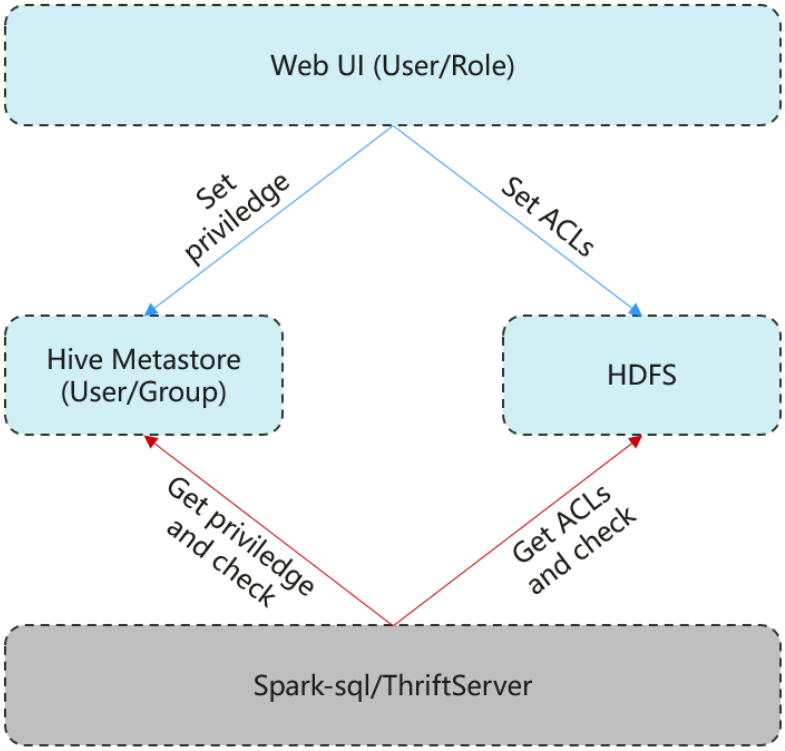
Spark SQL permission management indicates the permission system for managing and controlling users' operations on databases, to ensure that different users can operate databases separately and securely. A user can operate another user's tables and databases only with the corresponding permissions. Otherwise, operations will be rejected.

Spark SQL permission management integrates the functions of Hive management. The Metastore service of Hive and the permission assignment function are required to enable Spark SQL permission management.

Figure 24-1 shows the basic architecture of Spark SQL permission management. This architecture includes two parts: granting permissions on the page, and obtaining and judging a service.

- Granting permissions on the page: Spark SQL only supports granting permissions on the page. On FusionInsight Manager, choose **System** > **Permission** to add or delete a user, user group, or a role, and to grant permissions or cancel permissions.
- Obtaining and judging a service: When the DDL and DML commands are received from a client, Spark SQL will obtain the client's permissions on database information from MetaStore, and check whether the required permissions are included. If the required permissions are included, continue the execution. If the required permissions are not included, reject the user's operations. After the MetaStore permissions are checked, ACL permission also needs to be checked on HDFS.

Figure 24-1 Spark SQL permission management architecture



Additionally, Spark SQL provides column and view permissions to meet requirements of different scenarios.

- Column permission

  Spark SQL permission control consists of metadata permission control and HDFS ACL permission control. When Hive MetaStore automatically synchronizes table permissions to the HDFS ACL, column-level permissions are not synchronized. In other words, a user with partial or all column-level permissions cannot access the entire HDFS file using the HDFS client.

  - In **spark-sql** mode, users with only column-level permissions cannot access HDFS files. Therefore, they cannot access the columns of the corresponding tables.
  - In Beeline/JDBCServer mode, permissions are assigned among users, for example, the permissions on the table created by user A are assigned to user B.

    - **hive.server2.enable.doAs**=**true** (configured in the **hive-site.xml** file on the Spark server)

      In this case, user B cannot query the information. You need to manually assign the read permission on the file in HDFS.

    - **hive.server2.enable.doAs**=false
      - Users A and B are connected by Beeline. User B can query the information.
      - User A creates a table using SQL statements, and user B can query the table in Beeline.

      However, information query is not supported in other scenarios, for example, user A uses Beeline to create a table and user B uses SQL

to query the table, or user A uses SQL to create a table and user B uses SQL to query the table. You need to manually assign the read permission on the file in HDFS.

NOTE

The **spark** user is the Spark administrator in HDFS ACL permission control. The permission control of the Beeline client user depends only on the metadata permission on Spark.

- View permission

  View permission indicates the operation permission such as query and modification on the view of a table, regardless of the corresponding permission of a table. Namely, if you have the permission to query the view of a table, the permission to query the table is not mandatory. The view permission is applicable to the whole table but not to the columns.

  Restrictions of view and column permissions on SparkSQL are similar. The following uses the view permission as an example:

  - In spark-sql mode, if you have only the view permission but not the table permission and do not have the permission to read HDFS, you cannot access the table data stored in HDFS. That is, you cannot query the view of the table.
  - In Beeline/JDBCServer mode, permissions are assigned among users, for example, the permissions on the view created by user A are assigned to user B.

    - **hive.server2.enable.doAs**=**true** (configured in the **hive-site.xml** file on the Spark server)

      In this case, user B cannot query the information. You need to manually assign the read permission on the file in HDFS.

    - **hive.server2.enable.doAs**=false
      - Users A and B are connected by Beeline. User B can query the information.
      - User A creates a view using SQL statements, and user B can query the view in Beeline.

      However, information query is not supported in other scenarios. For example, user A uses Beeline to create a view but user B cannot use SQL to query the view, or user A uses SQL to create a view but user B cannot use SQL to query the view. You need to manually assign the read permission on the file in HDFS.

  Permission of operations on the view of a table is as follows:

  - To create a view, you must have the CREATE permission on the database and the SELECT and SELECT_of_GRANT permissions on the tables.
  - Creating and describing a view only entail the SELECT permission on the view. Querying views and tables at the same time entails the SELECT permission on other tables. For example, to perform **select * from v1 join t1**, you must have the SELECT permission on the **v1** view and **t1** table, even through the **v1** view depends on the **t1** table.

NOTE

In Beeline/JDBCServer mode, to query a view, you must have the SELECT permission on the tables. In spark-sql mode, to query a view, you must have the SELECT permission on the view and tables.

- Deleting and modifying a view entail the permission of owner on the view.

SparkSQL Permission Model

If you want to perform SQL operations using SparkSQL, you must be granted with permissions of SparkSQL databases and tables (include external tables and views). The complete permission model of SparkSQL consists of the meta data permission and HDFS file permission. Permissions required to use a database or a table is just one type of SparkSQL permission.

- Metadata permissions

  Metadata permissions are controlled at the metadata layer. Similar to traditional relational databases, SparkSQL databases involve the CREATE and SELECT permissions, and tables and columns involve the SELECT, INSERT, UPDATE, and DELETE permissions. SparkSQL also supports the permissions of **OWNERSHIP** and **ADMIN**.

- Data file permissions (that is, HDFS file permissions)

  SparkSQL database and table files are stored in HDFS. The created databases or tables are saved in the **/user/hive/warehouse** directory of HDFS by default. The system automatically creates subdirectories named after database names and database table names. To access a database or table, you must have the **Read**, **Write** and **Execute** permissions on the corresponding file in HDFS.

To perform various operations on SparkSQL databases or tables, you need to associate the metadata permission and HDFS file permission. For example, to query SparkSQL data tables, you need to associate the metadata permission **SELECT** and HDFS file permissions **Read** and **Execute**.

Using the management function of Manager GUI to manage the permissions of SparkSQL databases and tables, only requires the configuration of metadata permission, and the system will automatically associate and configure the HDFS file permission. In this way, operations on the interface are simplified, and the efficiency is improved.

Usage Scenarios and Related Permissions

Creating a database with SparkSQL service requires users to join in the hive group, without granting a role. Users have all permissions on the databases or tables created by themselves in Hive or HDFS. They can create tables, select, delete, insert, or update data, and grant permissions to other users to allow them to access the tables and corresponding HDFS directories and files.

A user can access the tables or database only with permissions. Users' permissions vary depending on different SparkSQL scenarios.

Table 24-1 SparkSQL scenarios

Typical Scenario	Required Permission
Using SparkSQL tables, columns, or databases	Permissions required in different scenarios are as follows: • To create a table, the CREATE permission is required. • To query data, the SELECT permission is required. • To insert data, the INSERT permission is required.
Associating and using other components	In some scenarios, except the SparkSQL permission, other permissions may be also required. For example: Using Spark on HBase to query HBase data in SparkSQL requires HBase permissions.

In some special SparkSQL scenarios, other permissions must be configured separately.

Table 24-2 SparkSQL scenarios and required permissions

Scenario	Required Permission
Creating SparkSQL databases, tables, and external tables, or adding partitions to created Hive tables or external tables when data files specified by Hive users are saved to other HDFS directories except **/user/hive/warehouse**	• The directory must exist, the client user must be the owner of the directory, and the user must have the **Read**, **Write**, and **Execute** permissions on the directory. The user must have the **Read** and **Execute** permissions of all the upper-layer directories of the directory. • If the Spark version is later than 2, the **Create** permission of the Hive database is required if you want to create a HBase table. However, in Spark 1.5, the **Create** permissions of both the Hive database and HBase namespace are required if you want to create a HBase table.

Scenario	Required Permission
Importing all the files or specified files in a specified directory to the table using load	• The data source is a Linux local disk, the specified directory exists, and the system user **omm** has read and execute permission of the directory and all its upper-layer directories. The specified file exists, and user **omm** has the **Read** permission on the file and has the **Read** and **Execute** permissions on all the upper-layer directories of the file. • The data source is HDFS, the specified directory exists, and the SparkSQL user is the owner of the directory and has the **Read**, **Write**, and **Execute** permissions on the directory and its subdirectories, and has the **Read** and **Execute** permissions on all its upper-layer directories. The specified file exists, and the SparkSQL user is the owner of the file and has the **Read**, **Write**, and **Execute** permissions on the file and has the **Read** and **Execute** permissions on all its upper-layer directories.
Creating or deleting functions or modifying any database	The **ADMIN** permission is required.
Performing operations on all databases and tables in Hive	The user must be added to the **supergroup** user group, and be assigned the **ADMIN** permission.
After assigning the **Insert** permission on some DataSource tables, assigning the **Write** permission on table directories in HDFS before performing the insert or analyze operation	When the **Insert** permission is assigned to the **spark datasource** table, if the table format is text, CSV, JSON, Parquet, or ORC, the permission on the table directory is not changed. After the **Insert** permission is assigned to the DataSource table of the preceding formats, you need to assign the **Write** permission to the table directories in HDFS separately so that users can perform the insert or analyze operation on the tables.

24.2.2 Creating a Spark SQL Role

Scenario

This section describes how to create and configure a SparkSQL role on Manager. The Spark SQL role can be configured with the Spark administrator permission or the permission of performing operations on the table data.

Creating a database with Hive requires users to join in the **hive** group, without granting a role. Users have all permissions on the databases or tables created by themselves in Hive or HDFS. They can create tables, select, delete, insert, or update data, and grant permissions to other users to allow them to access the tables and corresponding HDFS directories and files. The created databases or tables are saved in the **/user/hive/warehouse** directory of HDFS by default.

NOTE

- If the current component uses Ranger for permission control, you need to configure permission management policies based on Ranger. For details, see **Adding a Ranger Access Permission Policy for Spark2x**.
- After Ranger authentication is enabled or disabled on Spark2x, you need to restart Spark2x and download the client again or update the client configuration file **spark/conf/spark-defaults.conf**.

  Enable Ranger authentication: **spark.ranger.plugin.authorization.enable=true**

  Disable Ranger authentication: **spark.ranger.plugin.authorization.enable=false**

Procedure

1. Log in to Manager, and choose **System > Permission > Role**.
2. Click **Create Role** and set a role name and enter description.
3. Set **Configure Resource Permission**. For details, see **Table 24-3**.
   - **Hive Admin Privilege**: Hive administrator permissions.
   - **Hive Read Write Privileges**: Hive data table management permission, which is the operation permission to set and manage the data of created tables.

     NOTE

     - Hive role management supports Hive administrator permissions and the permissions to access tables and views, but does not support granting permissions on databases.
     - The permissions of the Hive administrator do not include the permission to manage HDFS.
     - If there are too many tables in the database or too many files in tables, the permission granting may last a while. For example, if a table contains 10,000 files, the permission granting lasts about 2 minutes.

Table 24-3 Setting a role

Task	Operation
Hive administrator permission	In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Hive** and select **Hive Admin Privilege**. After being bound to the Hive administrator role, perform the following operations during each maintenance operation: 1. Log in to the node where the Spark2x client is installed as the client installation user. 2. Run the following command to configure environment variables: For example, if the Spark2x client installation directory is **/opt/client**, run **source /opt/client/bigdata_env**. **source /opt/client/Spark2x/component_env** 3. Run the following command to perform user authentication: ***kinit*** *Hive service user* 4. Run the following command to log in to the client tool: **/opt/client/Spark2x/spark/bin/beeline -u "jdbc:hive2://<zkNode1_IP>:<zkNode1_Port>,<zkNode2_IP>:<zkNode2_Port>,<zkNode3_IP>:<zkNode3_Port>/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=sparkthriftserver2x;user.principal=spark2x/hadoop.***<*System domain name*>**@**<*System domain name*>**;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.***<*System domain name*>**@**<*System domain name*>**;"**

Task	Operation
	NOTE • **<zkNode1_IP>:<zkNode1_Port>,<zkNode2_IP>:<zkNode2_Port>,<zkNode3_IP>:<zkNode3_Port>** indicates the ZooKeeper URL, for example, 192.168.81.37:2181,192.168.195.232:2181,192.168.169.84:2181. • **sparkthriftserver** indicates a ZooKeeper directory, from which a random TriftServer or ProxyThriftServer is connected by the client. • You can log in to Manager, choose **System** > **Permission** > **Domain and Mutual Trust**, and view the value of **Local Domain**, which is the current system domain name. **spark2x/hadoop.**<*System domain name*> is the username. All letters in the system domain name contained in the username are lowercase letters. For example, **Local Domain** is set to **9427068F-6EFA-4833-B43E-60CB641E5B6C.COM**, and the username is **spark2x/hadoo.9427068f-6efa-4833-b43e-60cb641e5b6c.com**. 5. Run the following command to update the administrator permissions: **set role admin;**
Setting the permission to query a table of another user in the default database	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Hive** > **Hive Read Write Privileges**. 2. Click the name of the specified database in the database list. Tables in the database are displayed. 3. In the **Permission** column of the specified table, select **SELECT**.
Setting the permission to import data to a table of another user in the default database	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Hive** > **Hive Read Write Privileges**. 2. Click the name of the specified database in the database list. Tables in the database are displayed. 3. In the **Permission** column of the specified table, select **DELETE** and **INSERT**.

4. Click **OK**.

24.2.3 Configuring User Permissions for Spark Tables, Columns, and Databases

Scenario

You can configure related permissions if you need to access tables or databases created by other users. SparkSQL supports column-based permission control. If a user needs to access some columns in tables created by other users, the user must be granted the permission for columns. The following describes how to grant table, column, and database permissions to users by using the role management function of Manager.

Procedure

The operations for granting permissions on SparkSQL tables, columns, and databases are the same as those for Hive. For details, see **Hive User Permission Management**.

NOTE

- Any permission for a table in the database is automatically associated with the HDFS permission for the database directory to facilitate permission management. When any permission for a table is canceled, the system does not automatically cancel the HDFS permission for the database directory to ensure performance. In this case, users can only log in to the database and view table names.
- If you add or remove query permission for a database role, the corresponding query permission for tables in that database will also be added or removed automatically. This mechanism is inherited from Hive.
- In Spark, the column name of the struct data type cannot contain special characters, that is, characters other than letters, digits, and underscores (_). If the column name of the struct data type contains special characters, the column cannot be displayed on the FusionInsight Manager console when you grant permissions to roles on the role page.

Concepts

SparkSQL statements are processed in SparkSQL. **Table 24-4** describes the permission requirements.

Table 24-4 Scenarios of using SparkSQL tables, columns, or databases

Scenario	Required Permission
CREATE TABLE	**CREATE**, RWX+ownership (for creating external tables - the location) **NOTE** When creating datasource tables in a specified file path, the RWX and ownership permission on the file next to the path is required.
DROP TABLE	**Ownership** (of table)
DROP TABLE PROPERTIES	**Ownership**
DESCRIBE TABLE	**Select**

Scenario	Required Permission
SHOW PARTITIONS	**Select**
ALTER TABLE LOCATION	**Ownership**, RWX+ownership (for new location)
ALTER PARTITION LOCATION	**Ownership**, RWX+ownership (for new partition location)
ALTER TABLE ADD PARTITION	**Insert**, RWX and ownership (for partition location)
ALTER TABLE DROP PARTITION	**Delete**
ALTER TABLE(all of them except the ones above)	**Update**,**Ownership**
TRUNCATE TABLE	**Ownership**
CREATE VIEW	**Select**, **Grant Of Select**, and **CREATE**
ALTER VIEW PROPERTIES	**Ownership**
ALTER VIEW RENAME	**Ownership**
ALTER VIEW ADD PARTS	**Ownership**
ALTER VIEW AS	**Ownership**
ALTER VIEW DROPPARTS	**Ownership**
ANALYZE TABLE	**Search**, **Insert**
SHOW COLUMNS	**Select**
SHOW TABLE PROPERTIES	**Select**
CREATE TABLE AS SELECT	**Select**, **CREATE**
SELECT	**Select** **NOTE** The same as tables, you need to have the **Select** permission on a view when performing a SELECT operation on the view.
INSERT	**Insert**, **Delete (for overwrite)**
LOAD	**Insert**, **Delete**, RWX+ownership(input location)
SHOW CREATE TABLE	**Select** and **Grant Of Select**
CREATE FUNCTION	**ADMIN**
DROP FUNCTION	**ADMIN**
DESC FUNCTION	-
SHOW FUNCTIONS	-

Scenario	Required Permission
MSCK (metastore check)	**Ownership**
ALTER DATABASE	**ADMIN**
CREATE DATABASE	-
SHOW DATABASES	-
EXPLAIN	**Select**
DROP DATABASE	**Ownership**
DESC DATABASE	-
CACHE TABLE	**Select**
UNCACHE TABLE	**Select**
CLEAR CACHE TABLE	**ADMIN**
REFRESH TABLE	**Select**
ADD FILE	**ADMIN**
ADD JAR	**ADMIN**
HEALTHCHECK	-

24.2.4 Configuring Permissions for Spark SQL Service User

Scenario

Spark SQL may need to be associated with other components. For example, Spark on HBase requires HBase permissions. The following describes how to associate Spark SQL with HBase.

Prerequisites

- The Spark client has been installed in a directory, for example, **/opt/client**.
- You have obtained a user account with the MRS cluster administrator permissions, for example, **admin**.

Procedure

- **Spark on HBase authorization**

  After the permissions are assigned, you can use statements that are similar to SQL statements to access HBase tables from Spark SQL. The following uses the procedure for assigning a user the permissions to query HBase tables as an example.

  NOTE

  Set **spark.yarn.security.credentials.hbase.enabled** to **true**.

a. On Manager, create a role, for example, **hive_hbase_create**, and grant the permission to create HBase tables to the role.

   In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **HBase** > **HBase Scope** > **global**. Select **create** of the namespace **default**, and click **OK**.

   NOTE

   In this example, the created table is saved in the default database of Hive and has the CREATE permission of the default database. If you save the table to a Hive database other than **default**, perform the following operations:

   In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Hive** > **Hive Read Write Privileges**, select **CREATE** for the desired database, and click **OK**.

b. On Manager, create a role, for example, **hive_hbase_submit**, and grant the permission to submit tasks to the Yarn queue.

   In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Yarn** > **Scheduling Queue** > **root**. Select **Submit** of **default**, and click **OK**.

c. On Manager, create a human-machine user, for example, **hbase_creates_user**, add the user to the **hive** group, and bind the **hive_hbase_create** and **hive_hbase_submit** roles to create Spark SQL and HBase tables.

d. Log in to the node where the client is installed as the client installation user.

e. Run the following command to configure environment variables:

   source /opt/client/bigdata_env

   source /opt/client/Spark2x/component_env

f. Run the following command to authenticate the user:

   kinit hbase_creates_user

g. Run the following commands to enter the shell environment on the Spark JDBCServer client:

   /opt/client/Spark2x/spark/bin/beeline -u "jdbc:hive2://<zkNode1_IP>:<zkNode1_Port>,<zkNode2_IP>:<zkNode2_Port>,<zkNode3_IP>:<zkNode3_Port>/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=sparkthriftserver2x;user.principal=spark2x/hadoop.*<System domain name>***@***<System domain name>***;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.***<System domain name>***@***<System domain name>***;"**

h. Run the following command to create a table in Spark SQL and HBase, for example, create the **hbaseTable** table:

   create table hbaseTable (id string, name string, age int) using org.apache.spark.sql.hbase.HBaseSource options (hbaseTableName "table1", keyCols "id", colsMapping = ", name=cf1.cq1, age=cf1.cq2");

   The created Spark SQL table and the HBase table are stored in the Hive database **default** and the HBase namespace **default**, respectively.

i. On Manager, create a role, for example, **hive_hbase_select**, and grant the role the permission to query Spark SQL on HBase table **hbaseTable** and HBase table **hbaseTable**.

- In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **HBase** > **HBase Scope** > **global** > **default**. Select **read** for the **hbaseTable** table, and click **OK** to grant the table query permission to the HBase role.
- Edit the role. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **HBase** > **HBase Scope** > **global** > **hbase**. Select **Execute** for **hbase:meta**, and click **OK**.
- Edit the role. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Hive** > **Hive Read Write Privileges** > **default**. Select **SELECT** for the **hbaseTable** table, and click **OK**.

j. On Manager, create a human-machine user, for example, **hbase_select_user**, add the user to the **hive** group, and bind the **hive_hbase_select** role to the user for querying Spark SQL and HBase tables.

k. Run the following command to configure environment variables:

source /opt/client/bigdata_env

source /opt/client/Spark2x/component_env

l. Run the following command to authenticate users:

kinit hbase_select_user

m. Run the following commands to enter the shell environment on the Spark JDBCServer client:

/opt/client/Spark2x/spark/bin/beeline -u "jdbc:hive2://<zkNode1_IP>:<zkNode1_Port>,<zkNode2_IP>:<zkNode2_Port>,<zkNode3_IP>:<zkNode3_Port>/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=sparkthriftserver2x;user.principal=spark2x/hadoop.*<System domain name>***@***<System domain name>***;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.***<System domain name>***@***<System domain name>***;"**

n. Run the following command to use a Spark SQL statement to query HBase table data:

select * from hbaseTable;

24.2.5 Configuring Spark Web UI ACLs

Scenario

Users need to implement security protection for Spark2x web UI when some data on the UI cannot be viewed by other users. Once a user attempts to log in to the UI, Spark2x can check the view ACL of the user to determine whether to allow the access.

Spark2x has two types of web UI. One is for running tasks. You can access the web UI using the application link on the native Yarn page or the REST APIs. The other one is for ended tasks. You can access the web UI using the Spark2x JobHistory service or the REST APIs.

NOTE

This section applies only to clusters in security mode (with Kerberos authentication enabled).

- Configuring the ACL of the web UI for running tasks

  For a running task, you can set the following parameters on the server:

  - **spark.admin.acls**: specifies the web UI administrator list.
  - **spark.admin.acls.groups**: specifies the administrator group list.
  - **spark.ui.view.acls**: specifies the Yarn page visitor list.
  - **spark.modify.acls.groups**: specifies the Yarn page visitor group list.
  - **spark.modify.acls**: specifies the web UI modifier list.
  - **spark.ui.view.acls.groups**: specifies the web UI modifier group list.

- Configuring the ACL of the web UI for ended tasks

  For ended tasks, use client parameter **spark.history.ui.acls.enable** to enable or disable the ACL access permission.

  If ACL control is enabled, configure client parameters **spark.admin.acls** and **spark.admin.acls.groups** to specify the web UI administrator list and administrator group list. Use client parameters **spark.ui.view.acls** and **spark.modify.acls.groups** to specify the visitor list and visitor group list that view web UI task details. Use client parameters **spark.modify.acls** and **spark.ui.view.acls.groups** to specify the visitor list and group list that modify web UI task details.

Configuration

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Spark2x**. Click **Configurations**, click **All Configurations**, search for **acl**, and modify the following parameters on the JobHistory, JDBCServer, SparkResource, and Spark pages:

Table 24-5 Parameter description

Parameter	Description	Default Value
spark.history.ui.acls.enable	Indicates whether JobHistory supports the permission verification of a single task.	true
spark.acls.enable	Indicates whether to enable Spark permission management. If this function is enabled, the system checks whether the user has the permission to access and modify task information.	true
spark.admin.acls	Indicates the list of Spark administrators who have the authority to manage all Spark tasks. You can configure multiple administrators and differentiate them by using commas (,) to separate them.	admin

Parameter	Description	Default Value
spark.admin.acls.groups	Indicates the list of Spark administrator groups that have the authority to manage all Spark tasks. You can configure multiple administrators and differentiate them by using commas (,) to separate them.	-
spark.modify.acls	Indicates the list of members who have the permission to modify Spark tasks. By default, the user who starts a task has the permission to modify the task. You can configure multiple users and separate them from each other using commas (,).	-
spark.modify.acls.groups	Indicates the list of groups that have the permission to modify Spark tasks. You can configure multiple groups and separate them from each other using commas (,).	-
spark.ui.view.acls	Indicates the list of members that have the permission to access Spark tasks. By default, the user who starts a task has the permission to modify the task. You can configure multiple users and separate them from each other using commas (,).	-
spark.ui.view.acls.groups	Indicates the list of groups that have the permission to access Spark tasks. You can configure multiple groups and separate them from each other using commas (,).	-

NOTE

If you use a client to submit tasks, you must download the client again after modifying the **spark.admin.acls**, **spark.admin.acls.groups**, **spark.modify.acls**, **spark.modify.acls.groups**, **spark.ui.view.acls**, and **spark.ui.view.acls.groups** parameters.

24.2.6 Permission Parameters of the Spark Client and Server

This section describes how to configure SparkSQL permission management functions (client configuration is similar to server configuration). To enable table permission, add following configurations on the client and server:

- **spark-defaults.conf** configuration file

Table 24-6 Parameter description (1)

Parameter	Description	Default Value
spark.sql.authorization.enabled	Specifies whether to enable permission authentication of the datasource statement. It is recommended that the parameter value be set to **true** to enable permission authentication.	true

- **hive-site.xml** configuration file

Table 24-7 Parameter description (2)

Parameter	Description	Default Value
hive.metastore.uris	Specifies the MetaStore service address of the Hive component, for example, **thrift://10.10.169.84:21088,thrift://10.10.81.37:21088**.	-
hive.metastore.sasl.enabled	Specifies whether the MetaStore service uses SASL to improve security. The table permission function must be enabled.	true
hive.metastore.kerberos.principal	Specifies the principal of the MetaStore service in the Hive component, for example, **hive/hadoop.**<*system domain name*>@<*system domain name*>.	hive-metastore/_HOST@EXAMPLE.COM
hive.metastore.thrift.sasl.qop	After the SparkSQL permission management function is enabled, set the parameter to **auth-conf**.	auth-conf
hive.metastore.token.signature	Specifies the token identifier of the MetaStore service, which is set to **HiveServer2ImpersonationToken**.	HiveServer2ImpersonationToken
hive.security.authenticator.manager	Specifies the manager authenticated by the Hive client, which is set to **org.apache.hadoop.hive.ql.security.SessionStateUserGroupAuthenticator**.	org.apache.hadoop.hive.ql.security.SessionStateUserMSGroupAuthenticator
hive.security.authorization.enabled	Specifies whether to enable client authentication, which is set to **true**.	true

Parameter	Description	Default Value
hive.security.authorization.createtable.owner.grants	Specifies which permissions are granted to the owner who creates the table, which is set to **ALL**.	ALL

- **core-site.xml** configuration file of the MetaStore service

Table 24-8 Parameter description (3)

Parameter	Description	Default Value
hadoop.proxyuser.spark.hosts	Specifies the hosts from which Spark users can be masqueraded, which is set to *, indicating all hosts.	-
hadoop.proxyuser.spark.groups	Specifies the user groups from which Spark users can be masqueraded, which is set to *, indicating all user groups.	-

24.3 Using the Spark Client

This section describes how to use Spark2x to submit Spark applications, including Spark Core and Spark SQL. Spark Core is the kernel module of Spark. It executes tasks and is used to compile Spark applications. Spark SQL is a module that executes SQL statements.

Scenario Description

Develop a Spark application to perform the following operations on logs about netizens' dwell time for online shopping on a weekend.

- Collect statistics on female netizens who dwell on online shopping for more than 2 hours on the weekend.
- The first column in the log file records names, the second column records genders, and the third column records the dwell durations in the unit of minute. Three columns are separated by comma (,).

log1.txt: logs collected on Saturday

```
LiuYang,female,20
YuanJing,male,10
GuoYijun,male,5
CaiXuyu,female,50
Liyuan,male,20
FangBo,female,50
LiuYang,female,20
YuanJing,male,10
GuoYijun,male,50
```



```
CaiXuyu,female,50
FangBo,female,60
```

log2.txt: logs collected on Sunday

```
LiuYang,female,20
YuanJing,male,10
CaiXuyu,female,50
FangBo,female,50
GuoYijun,male,5
CaiXuyu,female,50
Liyuan,male,20
CaiXuyu,female,50
FangBo,female,50
LiuYang,female,20
YuanJing,male,10
FangBo,female,50
GuoYijun,male,50
CaiXuyu,female,50
FangBo,female,60
```

Prerequisites

- On Manager, you have created a user and granted the HDFS, Yarn, Kafka, and Hive permissions to the user.
- You have installed and configured tools such as IntelliJ IDEA and JDK based on the development language.
- You have installed the Spark2x client and configured the client network connection.
- For Spark SQL programs, you have started Spark SQL or Beeline on the client to enter SQL statements.

Procedure

Step 1 Obtain the sample project and import it to IDEA. Import the JAR package on which the sample project depends. Use IDEA to configure and generate JAR packages.

Step 2 Prepare the data required by the sample project.

Save the original log files in the scenario description to the HDFS system.

1. Create two text files (**input_data1.txt** and **input_data2.txt**) on the local host and copy the content in the **log1.txt** and **log2.txt** files to the **input_data1.txt** and **input_data2.txt** files, respectively.
2. Create the **/tmp/input** directory in HDFS, and upload **input_data1.txt** and **input_data2.txt** to the **/tmp/input** directory:

Step 3 Upload the generated JAR package to the Spark2x running environment (Spark2x client), for example, **/opt/female**.

Step 4 Go the client directory, configure the environment variables, and log in to the system. When you use a client to connect to a specific instance in a scenario where multiple Spark2x instances are installed or Spark and Spark2x instances are installed, run the following commands to load the environment variables of the instance.

source bigdata_env

source Spark2x/component_env

kinit <*Service user for authentication*>

Step 5 Run the following script in the **bin** directory to submit the Spark application:

spark-submit --class*com.huaweixxx.bigdata.spark.examples.FemaleInfoCollection***--master yarn-client***/opt/female/FemaleInfoCollection.jar <inputPath>**

NOTE

- **FemaleInfoCollection.jar** is the JAR package generated in **Step 1**.
- **<inputPath>** is the directory created in **Step 2.2**.

Step 6 (Optional) After calling the **spark-sql** or **spark-beeline** script in the **bin** directory, directly enter SQL statements to perform operations such as query.

For example, create a table, insert a piece of data, and then query the table.

```
spark-sql> CREATE TABLE TEST(NAME STRING, AGE INT);
Time taken: 0.348 seconds
spark-sql>INSERT INTO TEST VALUES('Jack', 20);
Time taken: 1.13 seconds
spark-sql> SELECT * FROM TEST;
Jack    20
Time taken: 0.18 seconds, Fetched 1 row(s)
```

Step 7 View the running result of the Spark application.

- View the running result data in a specified file.

  The storage path and format of the result data are specified by the Spark application.

- Check the running status on the web page.

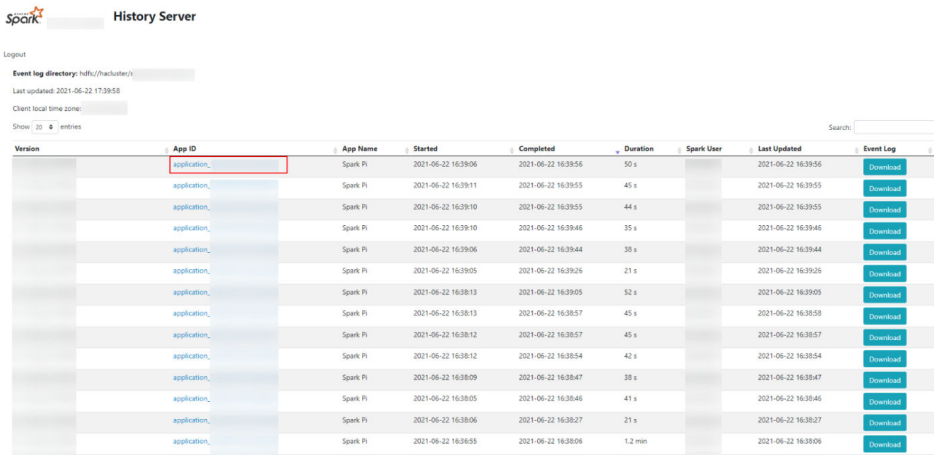
  a. Log in to Manager. Select **Spark2x** from the **Service** drop-down list.

  b. Go to the Spark2x overview page and click an instance in the Spark web UI, for example, **JobHistory2x(host2)**.

  c. The History Server UI is displayed.

  The History Server UI is used to display the status of Spark applications that are complete or incomplete.

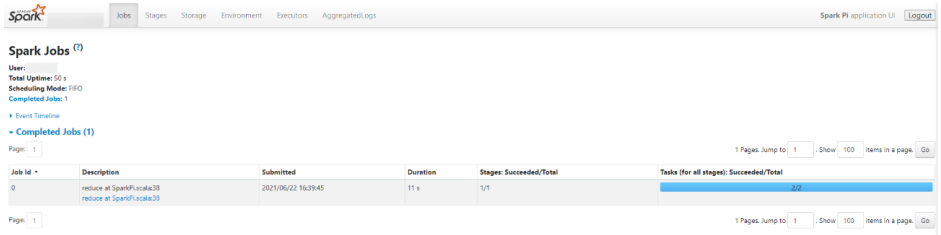
  Figure 24-2 History Server UI

  d. Select an application ID and click this page to go to the Spark UI of the application.

Spark UI: used to display the status of running applications.

Figure 24-3 Spark UI



- View Spark logs to learn application runtime conditions.

  View **Spark Log Overview** to learn application running status, and adjust applications based on log information.

----End

24.4 Accessing the Spark Web UI

Scenario

Once the Spark component is installed in the MRS cluster, you can easily monitor the status of Spark applications on the Spark web UI.

This section describes how to access the Spark web UI in an MRS cluster.

Prerequisites

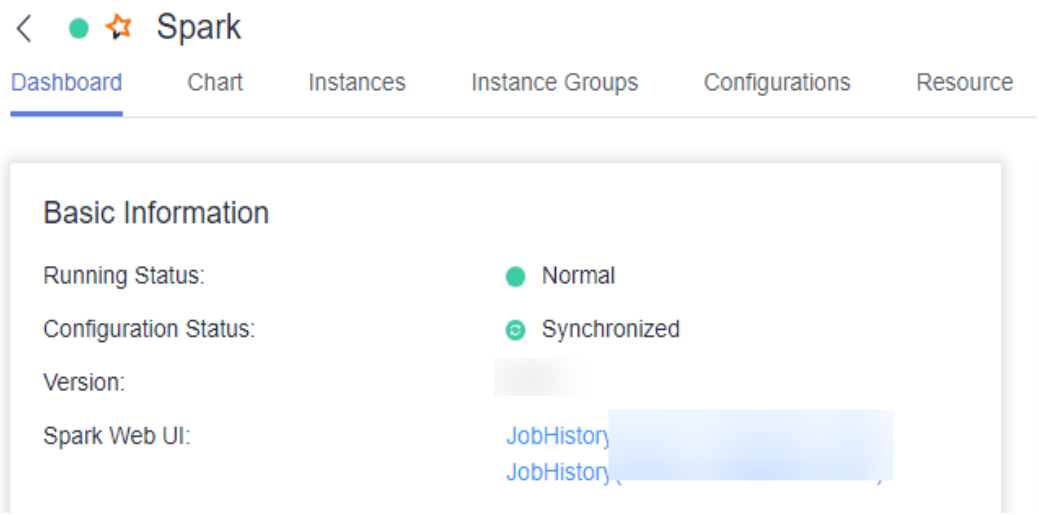
- The Spark component has been installed in an MRS cluster and is running properly.
- A user with the Spark management permission has been created. The **hadoop**, **hive**, and **supergroup** user groups have been added, and the **hadoop** user group has been added as the primary group.

Procedure

Step 1 Log in to Manager as a user with the Spark management permissions and choose **Cluster** > **Services** > **Spark**.

Step 2 On the Spark overview page, click **JobHistory(xxx)** next to the Spark web UI.

Figure 24-4 Spark Web UI



Step 3 Check the status of all Spark applications on the displayed Spark web UI.

The Spark web UI displays information such as the application ID, application name, start time, execution time, and user to whom the application belongs.

----End

24.5 Submitting a Spark Job as a Proxy User

NOTE

This section applies only to MRS 3.3.0 and later versions.

Scenario

Submit a Spark task as the actual running user or a proxy user. This section describes how to enable the proxy user function to submit Spark tasks.

Prerequisites

The **test** (actual running user) and **test1** (proxy user) users have been created and added to the **hadoop** (primary group), **hive**, and **supergroup** groups.

Submitting a Spark Task as a Proxy User in spark-beeline

Step 1 Modify JDBCServer instance parameters.

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Spark**. Click **Configurations**, click **All Configurations**, click **JDBCServer(Role)**, select **Customization**, find the **spark.core-site.customized.configs** parameter, and add the following custom parameters:

Parameter	Value
hadoop.proxyuser.*test*.hosts	*

Parameter	Value
hadoop.proxyuser.*test*.groups	*

NOTE

- In the configuration, **test** is the actual running user.
- The value of **hadoop.proxyuser.test.hosts** is *****, which indicates that any proxy user can be used after user **test** is connected and the number of cluster nodes is not limited.
- The value of **hadoop.proxyuser.test.groups** is *****, which indicates that any proxy user can be used after user **test** is connected and the proxy user can belong to any user group.

Step 2 Change the values of the following parameters to switch to the JDBCServer multi-instance mode:

Parameter	Value
spark.scheduler.allocation.file	#{conf_dir}/fairscheduler.xml
spark.thriftserver.proxy.enabled	false

Step 3 Save the configuration and restart the Spark service.

Step 4 Log in to the Spark client node and run the following commands:

cd *Client installation directory*

source bigdata_env

source Spark/component_env

In security mode, additionally run the following command:

kinit test Enter the password for authentication. (Change the password upon your first login.)

Step 5 Run the following Beeline commands of Spark to submit a task:

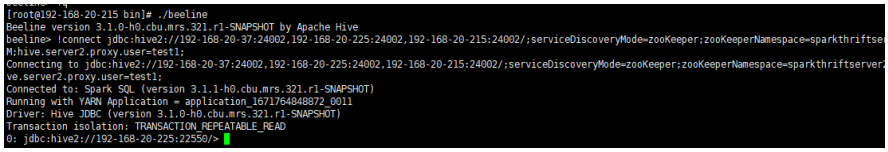
cd /opt/client/Spark/spark/bin

./beeline

!connect jdbc:hive2://*IP address of the node where the ZooKeeper instance is*:*Port number of the ZooKeeper client*,*IP address of the node where the ZooKeeper instance is*:*Port number of the ZooKeeper client*,*IP address of the node where the ZooKeeper instance is*:*Port number of the ZooKeeper client*/**;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=sparkthriftserver;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.hadoop.com@HADOOP.COM;hive.server2.proxy.user=***test1***

Specifically:

- *IP address of the node where the ZooKeeper instance is*: To obtain the IP address, on FusionInsight Manager, choose **Cluster** > **Services** > **ZooKeeper** and click **Instance**.
- *Port number of the ZooKeeper client*: To obtain the port number, on FusionInsight Manager, choose **Cluster** > **Services** > **ZooKeeper**, click **Configurations** then **All Configurations**, and search for **clientPort**.
- **hive.server2.proxy.user=test1**: **test1** is a proxy user.

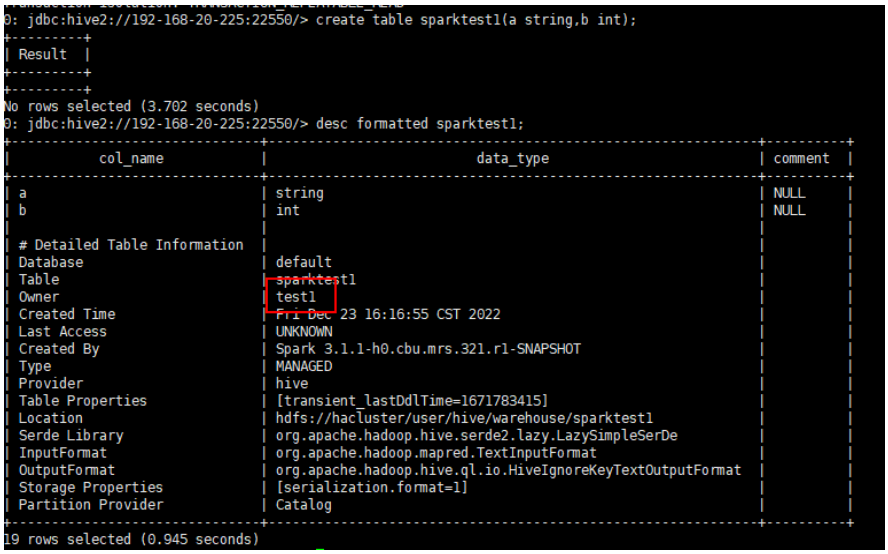


Step 6 Create a Spark table.

create table *sparktest1***(a string,b int);**

View the created table.

desc formatted *sparktest1*;



You can see that **Owner** of the table is the proxy user **test1**, and the proxy user is successfully used.

----End

Submitting a Spark Task as a Proxy User in spark-sql and spark-submit

Step 1 Modify the HDFS instance configuration. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS**. Click **Configurations** then **All Configurations**, click **HDFS(Service)**, select **Customization**, find the **hdfs.core-site.customized.configs** parameter, add the following custom parameters, and save the configuration:

Parameter	Value
hadoop.proxyuser.*test*.hosts	*

Parameter	Value
hadoop.proxyuser.*test*.groups	*

Step 2 Modify the Yarn instance configuration. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Yarn**. Click **Configurations** then **All Configurations**, click **Yarn(Service)**, select **Customization**, find the **yarn.core-site.customized.configs** parameter, add the following custom parameters, and save the configuration:

Parameter	Value
hadoop.proxyuser.*test*.hosts	*
hadoop.proxyuser.*test*.groups	*

Step 3 Modify the SparkReource instance configuration. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Spark**. Click **Configurations** then **All Configurations**, click **SparkResource(Role)**, select **Customization**, find the **spark.core-site.customized.configs** parameter, add the following custom parameters, and save the configuration:

Parameter	Value
hadoop.proxyuser.*test*.hosts	*
hadoop.proxyuser.*test*.groups	*

Step 4 Modify the Hive instance configuration. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Hive**. Click **Configurations** then **All Configurations**, click **Hive(Service)**, select **Customization**, find the **core.site.customized.configs** parameter, add the following custom parameters, and save the configuration:

Parameter	Value
hadoop.proxyuser.*test*.hosts	*
hadoop.proxyuser.*test*.groups	*

Step 5 Restart the HDFS, YARN, Spark, and Hive services and update the configuration files on their clients.

Step 6 Log in to the Spark client node and run the following commands:

cd *Client installation directory*

source bigdata_env

source Spark/component_env

In security mode, additionally run the following command:

kinit test Enter the password for authentication. (Change the password upon your first login.)

Step 7 Submit the spark-sql task.

spark-sql --master yarn --proxy-user *test1*

Step 8 Create a Spark table.

create table *sparktest2*(**a string,b int);**

View the created table.

desc formatted *sparktest2*;

```
spark-sql>
         > create table sparktest2(a string,b int);
2022-12-24 15:56:04,119 | WARN  | main | The enable mv value "null" is invalid. Using the de
2022-12-24 15:56:04,134 | WARN  | main | The value "LOCALLOCK" configured for key carbon.loc
igureLockType(CarbonProperties.java:444)
2022-12-24 15:56:05,493 | WARN  | main | A Hive serde table will be created as there is no t
rg.apache.spark.sql.catalyst.analysis.ResolveSessionCatalog.logWarning(Logging.scala:69)
Time taken: 2.845 seconds
spark-sql> desc formatted sparktest2;
a       string  NULL
b       int     NULL

# Detailed Table Information
Database        default
Table   sparktest2
Owner   test1
Created Time    Sat Dec 24 15:56:06 CST 2022
Last Access     UNKNOWN
Created By      Spark 3.1.1-h0.cbu.mrs.321.r1-SNAPSHOT
Type    MANAGED
Provider        hive
Table Properties        [transient_lastDdlTime=1671868566]
Location        hdfs://hacluster/user/hive/warehouse/sparktest2
Serde Library   org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat     org.apache.hadoop.mapred.TextInputFormat
OutputFormat    org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Storage Properties      [serialization.format=1]
Partition Provider      Catalog
Time taken: 0.859 seconds, Fetched 19 row(s)
spark-sql>
```

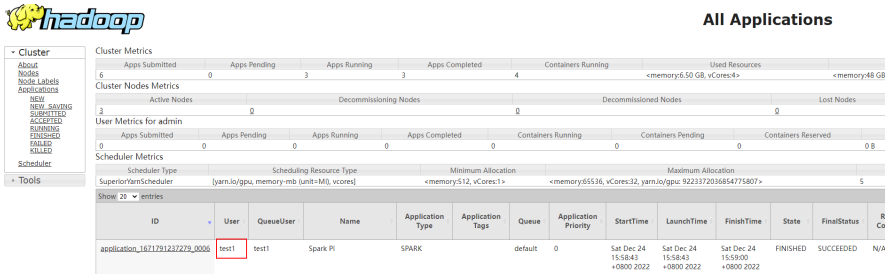
You can see that **Owner** of the table is the proxy user **test1**, and the proxy user is successfully used.

Step 9 Use the redelivered client to submit the spark-submit task.

spark-submit --master yarn --class org.apache.spark.examples.SparkPi --master yarn-client --proxy-user *test1* **/opt/client/Spark/spark/examples/jars/spark-examples_*.jar**

```
[root@192-168-20-215 conf]# spark-submit --master yarn --class org.apache.spark.examples.SparkPi --master yarn-client --proxy-user test1 ../examples/jars/spark-examples_*.jar
2022-12-24 15:58:37,540 | WARN  | main | The configuration key 'spark.yarn.access.hadoopFileSystems' has been deprecated as of Spark 3.0 and may be removed in the future. Please
ad. | org.apache.spark.SparkConf.logWarning(Logging.scala:69)
2022-12-24 15:58:37,542 | WARN  | main | The configuration key 'spark.yarn.kerberos.relogin.period' has been deprecated as of Spark 3.0 and may be removed in the future. Please u
pache.spark.SparkConf.logWarning(Logging.scala:69)
2022-12-24 15:58:37,543 | WARN  | main | The configuration key 'spark.executor.plugins' has been deprecated as of Spark 3.0.0 and may be removed in the future. Feature replaced w
spark.SparkConf.logWarning(Logging.scala:69)
2022-12-24 15:58:37,544 | WARN  | main | The configuration key 'spark.reducer.maxReqSizeShuffleToMem' has been deprecated as of Spark 2.3 and may be removed in the future. Please
```

Step 10 View information about applications running in YARN.



You can see that the running user of the task is **test1**, and the proxy user is successfully used.

----End

24.6 Configuring Spark to Read HBase Data

Scenario

Spark on HBase allows users to query HBase tables in Spark SQL and to store data for HBase tables by using the Beeline tool. You can use HBase APIs to create, read data from, and insert data into tables.

Spark On HBase

Step 1 Log in to Manager and choose **Cluster** > **Cluster Properties** to check whether the cluster is in security mode.

- If yes, go to **Step 2**.
- If no, go to **Step 5**.

Step 2 Choose **Cluster** > **Services** > **Spark2x**. Click **Configurations**, click **All Configurations**, click **JDBCServer2x**, select **Default**, and modify the following parameter:

Table 24-9 Parameter list 1

Parameter	Default Value	Changed To
spark.yarn.security.credentials.hbase.enabled	false	true

NOTE

To ensure that Spark2x can access HBase for a long time, do not modify the following parameters of the HBase and HDFS services:

- dfs.namenode.delegation.token.renew-interval
- dfs.namenode.delegation.token.max-lifetime
- hbase.auth.key.update.interval
- hbase.auth.token.max.lifetime (The value is fixed to **604800000** ms, that is, 7 days.)

If the preceding parameter configuration must be modified based on service requirements, ensure that the value of the HDFS parameter **dfs.namenode.delegation.token.renew-interval** is not greater than the values of the HBase parameters **hbase.auth.key.update.interval**, **hbase.auth.token.max.lifetime**, and **dfs.namenode.delegation.token.max-lifetime**.

Step 3 Choose **SparkResource2x** > **Default** and modify the following parameters.

Table 24-10 Parameter list 2

Parameter	Default Value	Changed To
spark.yarn.security.credentials.hbase.enabled	false	true

Step 4 Restart the Spark2x service for the configuration to take effect.

NOTE

To use the Spark on HBase function on the Spark2x client, you need to download and install the Spark2x client again.

Step 5 On the Spark2x client, use the spark-sql or spark-beeline connection to query tables created by Hive on HBase. You can create an HBase table by running SQL commands or create an external table to associate the HBase table. Before creating tables, ensure that HBase tables exist in HBase. The HBase table **table1** is used as an example.

1. Run the following commands to create the HBase table using the Beeline tool:

   create table *hbaseTable*
   (
   id string,
   name string,
   age int
)
   using org.apache.spark.sql.hbase.HBaseSource
   options(
   hbaseTableName *"table1"***,**
   keyCols *"id"***,**
   colsMapping "
   name=cf1.cq1,
   age=cf1.cq2
   ");

   NOTE

   - **hbaseTable**: name of the created Spark table
   - **id string,name string**, **age int**: field name and field type of the Spark table
   - **table1**: name of the HBase table
   - *id*: row key column name of the HBase table
   - *name=cf1.cq1*, *age=cf1.cq2*: mapping between columns in the Spark table and columns in the HBase table. The **name** column of the Spark table maps the **cq1** column in the **cf1** column family of the HBase table, and the **age** column of the Spark table maps the **cq2** column in the **cf1** column family of the HBase table.

2. Import data to the HBase table using a CSV file.

   hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator="," -

Dimporttsv.columns=HBASE_ROW_KEY,*cf1:cq1,cf1:cq2,cf1:cq3,cf1:cq4,cf1:cq5 table1 /hperson*

table1 indicates the name of the HBase table and **/hperson** indicates the path where the CSV file is stored.

3. Query data in spark-sql or spark-beeline. *hbaseTable* is the corresponding Spark table name. The command is as follows:

   select * from *hbaseTable;*

----End

Spark on HBaseV2

Step 1 Log in to Manager and choose **Cluster** > **Cluster Properties** to check whether the cluster is in security mode.

- If yes, go to **Step 2**.
- If no, go to **Step 5**.

Step 2 Click **Cluster** and click the name of the desired cluster. Choose **Service** > **Spark2x**, click **Configurations**, click **All Configurations**, and choose **JDBCServer2x** > **Default**. Modify the following parameter.

Table 24-11 Parameter list 1

Parameter	Default Value	Changed To
spark.yarn.security.credentials.hbase.enabled	false	true

NOTE

To ensure that Spark2x can access HBase for a long time, do not modify the following parameters of the HBase and HDFS services:

- dfs.namenode.delegation.token.renew-interval
- dfs.namenode.delegation.token.max-lifetime
- hbase.auth.key.update.interval
- hbase.auth.token.max.lifetime (The value is fixed to **604800000** ms, that is, 7 days.)

If the preceding parameter configuration must be modified based on service requirements, ensure that the value of the HDFS parameter **dfs.namenode.delegation.token.renew-interval** is not greater than the values of the HBase parameters **hbase.auth.key.update.interval**, **hbase.auth.token.max.lifetime**, and **dfs.namenode.delegation.token.max-lifetime**.

Step 3 Choose **SparkResource2x** > **Default** and modify the following parameters.

Table 24-12 Parameter list 2

Parameter	Default Value	Changed To
spark.yarn.security.credentials.hbase.enabled	false	true

Step 4 Restart the Spark2x service for the configuration to take effect.

NOTE

If you need to use the Spark on HBase function on the Spark2x client, download and install the Spark2x client again.

Step 5 On the Spark2x client, use the spark-sql or spark-beeline connection to query tables created by Hive on HBase. You can create an HBase table by running SQL commands or create an external table to associate the HBase table. For details, see the following description. The following uses the HBase table **table1** as an example.

1. Create a table using the spark-beeline tool.

   create table *hbaseTable*1

   (*id string, name string, age int***)**

   using org.apache.spark.sql.hbase.HBaseSourceV2

   options(

   hbaseTableName *"table2"***,**

   keyCols *"id"***,**

   colsMapping "*name=cf1.cq1,age=cf1.cq2***");**

   NOTE

   - **hbaseTable1**: name of the created Spark table
   - **id string,name string**, **age int**: field name and field type of the Spark table
   - **table2**: name of the HBase table
   - *id*: row key column name of the HBase table
   - *name=cf1.cq1*, *age=cf1.cq2*: mapping between columns in the Spark table and columns in the HBase table. The **name** column of the Spark table maps the **cq1** column in the **cf1** column family of the HBase table, and the **age** column of the Spark table maps the **cq2** column in the **cf1** column family of the HBase table.

2. Import data to the HBase table using a CSV file.

   hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator="," -Dimporttsv.columns=HBASE_ROW_KEY,*cf1:cq1,cf1:cq2,cf1:cq3,cf1:cq4,cf1:cq5 table2 /hperson*

   table2 indicates the name of the HBase table and **/hperson** indicates the path where the CSV file is stored.

3. Query data in spark-sql or spark-beeline. *hbaseTable1* indicates the corresponding Spark table name.

   select * from *hbaseTable1;*

----End

24.7 Configuring Spark Tasks Not to Obtain HBase Token Information

Scenario

When Spark is used to submit tasks, the driver obtains tokens from HBase by default. To access HBase, you need to configure the **jaas.conf** file for security authentication. If the **jaas.conf** file is not configured, the application will fail to run.

Therefore, perform the following operations based on whether the application involves HBase:

- If the application does not involve HBase, you do not need to obtain the HBase tokens. In this case, set **spark.yarn.security.credentials.hbase.enabled** to **false**.
- If the application involves HBase, set **spark.yarn.security.credentials.hbase.enabled** to **true** and configure the **jaas.conf** file on the driver as follows:

  ```
  {client}/spark/bin/spark-sql  --master yarn-client --principal {principal} --keytab {keytab} --driver-java-options "-Djava.security.auth.login.config={LocalPath}/jaas.conf"
  ```

  Specify Keytab and Principal in the **jaas.conf** file. The following is an example:

  ```
  Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  keyTab = "{LocalPath}/user.keytab"
  principal="super@<System domain name>"
  useTicketCache=false
  debug=false;
  };
  ```

Configuration

Configure the following parameter in the **spark-defaults.conf** file of the Spark client.

Table 24-13 Parameter description

Parameter	Description	Default Value
spark.yarn.security.credentials.hbase.enabled	Indicates whether HBase obtains a token. • **true**: HBase obtains a token. • **false**: HBase does not obtain a token.	false

24.8 Spark Core Enterprise-Class Enhancements

24.8.1 Configuring Spark HA to Enhance HA

24.8.1.1 Configuring Multi-active Instance Mode

Scenarios

In this mode, multiple ThriftServers coexist in the cluster and the client can randomly connect any ThriftServer to perform service operations. When one or multiple ThriftServers stop working, a client can connect to another functional ThriftServer.

Configuration Description

Log in to Manager and choose **Cluster** > **Services** > **Spark2x**. Click **Configurations**, click **All Configurations**, and search for and modify the following parameters:

Table 24-14 Parameter description

Parameter	Description	Default Value
spark.thriftserver.zookeeper.connection.timeout	Specifies the timeout interval of connection between ZooKeeper client and ThriftServer. The unit is millisecond.	60000
spark.thriftserver.zookeeper.session.timeout	Specifies the timeout interval of a ZooKeeper client session. The unit is millisecond.	90000
spark.thriftserver.zookeeper.retry.times	Specifies the retry times after ZooKeeper disconnection.	3
spark.yarn.queue	Specifies the Yarn queue where the JDBCServer service resides.	default

24.8.1.2 Configuring the Spark Multi-Tenant Mode

Scenarios

In multi-tenant mode, JDBCServers are bound with tenants. Each tenant corresponds to one or more JDBCServers, and a JDBCServer provides services for only one tenant. Different tenants can be configured with different Yarn queues to implement resource isolation.

Configuration Description

Log in to Manager and choose **Cluster** > **Services** > **Spark2x**. Click **Configurations**, click **All Configurations**, and search for and modify the following parameters:

Table 24-15 Parameter description

Parameter	Description	Default Value
spark.proxyserver.hash.enabled	Specifies whether to connect to ProxyServer using the Hash algorithm. • **true** indicates using the Hash algorithm. In multi-tenant mode, this parameter must be configured to **true**. • **false** indicates using random connection. In multi-active instance mode, this parameter must be configured to **false**.	true **NOTE** After this parameter is modified, you need to download the client again.
spark.thriftserver.proxy.enabled	Specifies whether to use the multi-tenant mode. • **false**: The multi-instance mode is used. • **true**: The multi-tenant mode is used.	true
spark.thriftserver.proxy.maxThriftServer-PerTenancy	Specifies the maximum number of JDBCServer instances that can be started by a tenant in multi-tenant mode.	1
spark.thriftserver.proxy.maxSessionPerThriftServer	Specifies the maximum number of sessions in a single JDBCServer instance in multi-tenant mode. If the number of sessions exceeds this value and the number of JDBCServer instances does not exceed the upper limit, a new JDBCServer instance is started. Otherwise, an alarm log is output.	50
spark.thriftserver.proxy.sessionWaitTime	Specifies the wait time before a JDBCServer instance is stopped when it has no session connections in multi-tenant mode.	180000
spark.thriftserver.proxy.sessionThreshold	In multi-tenant mode, when the session usage (formula: number of current sessions/ spark.thriftserver.proxy.maxSessionPerThriftServer x number of current JDBCServer instances) of the JDBCServer instance reaches the threshold, a new JDBCServer instance is automatically added.	100
spark.thriftserver.proxy.healthcheck.period	Specifies the period of JDBCServer health checks conducted by the JDBCServer proxy in multi-tenant mode.	60000

Parameter	Description	Default Value
spark.thriftserver.proxy.healthcheck.recheckTimes	Specifies the number of JDBCServer health check retries conducted by the JDBCServer proxy in multi-tenant mode.	3
spark.thriftserver.proxy.healthcheck.waitTime	Specifies the wait time for JDBCServer to respond to a health check request sent by the JDBCServer proxy.	10000
spark.thriftserver.proxy.session.check.interval	Specifies the period of JDBCServer proxy sessions in multi-tenant mode.	6h
spark.thriftserver.proxy.idle.session.timeout	Specifies the idle time interval of a JDBCServer proxy session in multi-tenant mode. If no operation is performed within this period, the session is closed.	7d
spark.thriftserver.proxy.idle.session.check.operation	Specifies whether to check that operations still exist on a JDBCServer proxy session when the session is checked for expiration in multi-tenant mode.	true
spark.thriftserver.proxy.idle.operation.timeout	Specifies the timeout interval of an operation in multi-tenant mode. An operation that times out is closed.	5d

24.8.1.3 Configuring the Switchover Between the Multi-active Instance Mode and the Multi-tenant Mode

Scenarios

When using a cluster, if you want to switch between multi-active instance mode and multi-tenant mode, the following configurations are required.

- Switch from multi-tenant mode to multi-active instance mode.

  Modify the following parameters of the Spark2x service:

  - spark.thriftserver.proxy.enabled=false
  - spark.scheduler.allocation.file=#{conf_dir}/fairscheduler.xml
  - spark.proxyserver.hash.enabled=false
- Switch from multi-active instance mode to multi-tenant mode.

  Modify the following parameters of the Spark2x service:

  - spark.thriftserver.proxy.enabled=true
  - spark.scheduler.allocation.file=./__spark_conf__/__hadoop_conf__/fairscheduler.xml

– spark.proxyserver.hash.enabled=true

Configuration Description

Log in to Manager and choose **Cluster** > **Services** > **Spark2x**. Click **Configurations**, click **All Configurations**, and search for and modify the following parameters:

Table 24-16 Parameter description

Parameter	Description	Default Value
spark.thriftserver.proxy.enabled	Specifies whether to use the multi-tenant mode. • **false**: The multi-instance mode is used. • **true**: The multi-tenant mode is used.	true
spark.scheduler.allocation.file	Specifies the fair scheduling file path. • If the multi-active instance mode is used, the path is changed to **#{conf_dir}/fairscheduler.xml**. • If multi-tenant mode is used, the path is changed to **./__spark_conf__/__hadoop_conf__/fairscheduler.xml**.	./__spark_conf__/__hadoop_conf__/fairscheduler.xml
spark.proxyserver.hash.enabled	Specifies whether to connect to ProxyServer using the Hash algorithm. • **true** indicates using the Hash algorithm. In multi-tenant mode, this parameter must be configured to **true**. • **false** indicates using random connection. In multi-active instance mode, this parameter must be configured to **false**.	true **NOTE** After this parameter is modified, you need to download the client again.

24.8.2 Configuring the Spark Native Engine

NOTE

This section applies only to MRS 3.3.0 or later.

Scenarios

The Spark Native engine uses the vectorized C++ acceleration library to accelerate Spark operators. Traditional SparkSQL is based on row data and uses JVM codegen to accelerate query. The JVM has a range of restrictions on the generated Java code, such as the method length and number of parameters, and the memory

bandwidth utilization of row data is low. The performance needs to be improved. When the mature vectorized C++ acceleration library is used, data is stored in the memory in vectorized format, which improves bandwidth utilization and speeds up queries by processing columns in batches.

You can enable the Spark Native engine to accelerate SparkSQL queries.

Constraints

- The Scan operator supports the following data types: Boolean, Integer, Long, Float, Double, String, Date, and Decimal.
- Parquet and ORC data formats are supported.
- OBS and HDFS file systems are supported.
- ADM64 and Arm architectures are supported.
- Spark SQL mode is supported.

Parameters

1. Modify the following parameters in the *Client installation directory*/**Spark/spark/conf/spark-defaults.conf** file on the Spark client.

Parameter	Description	Default Value
spark.plugins	Plug-in used by Spark. Set this parameter to **io.glutenproject.GlutenPlugin**. **NOTE** If **spark.plugins** has been configured, add **io.glutenproject.GlutenPlugin** to the file and separate them with commas (,).	N/A
spark.memory.offHeap.enabled	If this parameter is set to **true**, Native acceleration requires the off-heap memory of the JVM.	false
spark.memory.offHeap.size	Size of the off-heap memory. Set the value based on the site requirements. The initial value is 1 GB.	-1

Parameter	Description	Default Value
spark.yarn.dist.files	This parameter is used to distribute **libch.so** and **libjsig.so** to all nodes so that all executors can use the **spark.executorEnv.LD_PRELOAD** parameter to preload the above libraries. • For the x86 architecture, set this parameter to **{***Client installation directory***}/Spark/spark/native/libch.so,{***Client installation directory***}/JDK/jdk1.8.0_372/jre/lib/amd64/libjsig.so**. • For the Arm architecture, set this parameter to **{***Client installation directory***}/Spark/spark/native/libch.so,{***Client installation directory***}/JDK/jdk1.8.0_372/jre/lib/aarch64/libjsig.so**. **NOTE** If **spark.yarn.dist.files** has been configured, you can add this parameter to it and separate them with commas (,). **libch.so** and **libjsig.so** in the same path as **export LD_PRELOAD** in **spark-env.sh** in **2** must be used.	None
spark.executorEnv.LD_PRELOAD	Environment variable LD_PRELOAD for the executor. Set this parameter to **$PWD/libch.so $PWD/libjsig.so**. **NOTE** This parameter is used by the executor to preload **libch.so** and **libjsig.so**. If **spark.executorEnv.LD_PRELOAD** has been configured, add the preceding parameters and separate them with spaces.	None

Parameter	Description	Default Value
spark.gluten.sql.columnar.libpath	Path of the Native acceleration library on the server. This file does not exist if database mirroring is not used and is left empty.	Spark installation directory in the cluster, for example, **${BIGDATA_HOME}/FusionInsight_Spark_xxx/install/FusionInsight-Spark-*/spark/native/libch.so**.
spark.sql.orc.impl	**native**: The native ORC of Spark is used to read data. **hive**: Hive is used to process ORC data. Set this parameter to **native**.	hive
spark.gluten.sql.columnar.scanOnly	Whether to enable scanOnly for acceleration. Set this parameter to **true** to enable the scanOnly mode.	false

2. Modify the following parameters in the *Client installation directory*/**Spark/spark/conf/spark-env.sh** file on the Spark client.
   - For the x86 architecture:

     Set **export LD_PRELOAD** to *{Client installation directory}*/**Spark/spark/native/libch.so** *{Client installation directory}*/**JDK/jdk1.8.0_372/jre/lib/amd64/libjsig.so**.
   - For the Arm architecture:

     Set **export LD_PRELOAD** to *{Client installation directory}*/**Spark/spark/native/libch.so** *{Client installation directory}*/**JDK/jdk1.8.0_372/jre/lib/aarch64/libjsig.so**.

     Note: Use the **libch.so** and **libjsig.so** that are in the same path of the **spark.yarn.dist.files** parameter. If there are multiple SO files, separate them with commas (,) and add double quotation marks (") before and after each SO file.

24.8.3 Configuring the Size of the Spark Event Queue

Scenarios

Functions such as UI, EventLog, and dynamic resource scheduling in Spark are implemented through event transfer. Events include SparkListenerJobStart and SparkListenerJobEnd, which record each important process.

Each event is saved to a queue after it occurs. When creating a SparkContext object, Driver starts a thread to obtain an event from the queue in sequence and sends the event to each Listener. Each Listener processes the event after detecting the event.

Therefore, when the queuing speed is faster than the read speed, the queue overflows. As a result, the overflow event is lost, affecting the UI, EventLog, and dynamic resource scheduling functions. Therefore, a configuration item is added for more flexible use. You can set a proper value based on the memory size of the driver.

Configuration Description

Navigation path for setting parameters:

Before executing an application, modify the Spark service configuration. On Manager, choose **Cluster > Services** > **Spark2x** > **Configurations** and click **All Configurations**. Enter a parameter name in the search box.

Table 24-17 Parameter description

Parameter	Description	Default Value
spark.scheduler.listenerbus.eventqueue.capacity	Specifies the size of the event queue. Configure this parameter based on the memory of the driver.	1000000

NOTE

If the following information is displayed in the Driver log, the queue overflows.

1. Common application:
   ```
   Dropping SparkListenerEvent because no remaining room in event queue.
   This likely means one of the SparkListeners is too slow and cannot keep
   up with the rate at which tasks are being started by the scheduler.
   ```
2. Spark Streaming application:
   ```
   Dropping StreamingListenerEvent because no remaining room in event queue.
   This likely means one of the StreamingListeners is too slow and cannot keep
   up with the rate at which events are being started by the scheduler.
   ```

24.8.4 Configuring the Compression Format of a Parquet Table

Scenarios

The compression format of a Parquet table can be configured as follows:

1. If the Parquet table is a partitioned one, set the **parquet.compression** parameter of the Parquet table to specify the compression format. For example, set **tblproperties** in the table creation statement: **"parquet.compression"="snappy"**.
2. If the Parquet table is a non-partitioned one, set the **spark.sql.parquet.compression.codec** parameter to specify the compression format. The configuration of the **parquet.compression** parameter is invalid, because the value of the **spark.sql.parquet.compression.codec** parameter is read by the **parquet.compression** parameter. If the **spark.sql.parquet.compression.codec** parameter is not configured, the default value is **snappy** and will be read by the **parquet.compression** parameter.

Therefore, the **spark.sql.parquet.compression.codec** parameter can only be used to set the compression format of a non-partitioned Parquet table.

Configuration parameters

Navigation path for setting parameters:

On Manager, choose **Cluster> Services** > **Spark2x**. Click **Configurations**, click **All Configurations**, and enter a parameter name in the search box.

Table 24-18 Parameter description

Parameter	Description	Default Value
spark.sql.parquet.compression.codec	Used to set the compression format of a non-partitioned Parquet table.	snappy

24.8.5 Adapting to the Third-party JDK When Ranger Is Used

Scenarios

When Ranger is used as the permission management service of Spark SQL, the certificate in the cluster is required for accessing RangerAdmin. If you use a third-party JDK instead of the JDK or JRE in the cluster, RangerAdmin fails to be accessed. As a result, the Spark application fails to be started.

In this scenario, you need to perform the following operations to import the certificate in the cluster to the third-party JDK or JRE.

Configuration Method

Step 1 Run the following command to export the certificate from the cluster:

1. Install the cluster client. Assume that the installation path is **/opt/client**.
2. Run the following command to switch to the client installation directory:

   cd /opt/client
3. Run the following command to configure environment variables:

   source bigdata_env
4. Generate the certificate file.

   keytool -export -alias fusioninsightsubroot -storepass changeit -keystore /opt/client/JRE/jre/lib/security/cacerts -file fusioninsightsubroot.crt

Step 2 Import the certificate in the cluster to the third-party JDK or JRE.

Copy the **fusioninsightsubroot.crt** file generated in **Step 1** to the third-party JRE node, set the **JAVA_HOME** environment variable of the node, and run the following command to import the certificate:

keytool -import -trustcacerts -alias fusioninsightsubroot -storepass changeit -file fusioninsightsubroot.crt -keystore MY_JRE/lib/security/cacerts

NOTE

MY_JRE indicates the installation path of the third-party JRE. Change it based on the site requirements.

----End

24.8.6 Using the Spark Small File Combination Tool

NOTE

This section applies only to MRS 3.3.0 or later.

Scenarios

After the automatic small file merging feature is enabled, Spark writes data to the temporary directory and then checks whether the average file size of each partition is less than 16 MB (default value). If the average file size is less than 16 MB, the partition contains small files. Spark starts a job to merge these small files and writes the large files to the final table directory.

Constraints

- Only Hive and DataSource tables can be written.
- Parquet and ORC data formats are supported.

Parameters

Modify the following parameters in the *Client installation directory*/**Spark/spark/conf/spark-defaults.conf** file on the Spark client.

Parameter	Description	Default Value
spark.sql.mergeSmallFiles.enabled	If this parameter is set to **true**, Spark checks whether small files are written when writing data to the target table. If small files are found, Spark starts the file merging job.	false
spark.sql.mergeSmallFiles.threshold.avgSize	If the average file size of a partition is smaller than the value of this parameter, small file merging is started.	16MB
spark.sql.mergeSmallFiles.maxSizePerTask	Target size of each file after the merging.	256MB
spark.sql.mergeSmallFiles.moveParallelism	Maximum degree of parallelism of moving temporary files to the final directory. If the number of temporary files exceed the specified value, a file merging job is triggered.	10000

24.8.7 Using the Spark Small File Combination Tool

Tool Overview

In a large-scale Hadoop production cluster, HDFS metadata is stored in the NameNode memory, and the cluster scale is restricted by the memory limitation of each NameNode. If there are a large number of small files in the HDFS, a large amount of NameNode memory is consumed, which greatly reduces the read and write performance and prolongs the job running time. Based on the preceding information, the small file problem is a key factor that restricts the expansion of the Hadoop cluster.

This tool provides the following functions:

1. Checks the number of small files whose size is less than the threshold configured by the user in tables and returns the average size of all data files in the table directory.
2. Provides the function of combination table files. Users can set the average file size after combination.

Supported Table Types

Spark: Parquet, ORC, CSV, Text, and Json.

Hive: Parquet, ORC, CSV, Text, RCFile, Sequence and Bucket.

NOTE

1. After tables with compressed data are merged, Spark uses the default compression format Snappy for data compression. You can configure **spark.sql.parquet.compression.codec** (available values: **uncompressed**, **gzip**, and **snappy**) and **spark.sql.orc.compression.codec** (available values: **uncompressed**, **zlib**, **lzo**, and **snappy**) on the client to select the compression format for the Parquet and ORC tables. Compression formats available for Hive and Spark tables are different. Except the preceding compression formats, other compression formats are not supported.
2. To merge bucket table data, you need to add the following configurations to the **hive-site.xml** file on the Spark2x client:

```
<property>
<name>hive.enforce.bucketing</name>
<value>false</value>
</property>
<property>
<name>hive.enforce.sorting</name>
<value>false</value>
</property>
```

3. Spark does not support the feature of encrypting data columns in Hive.

Tool Usage

Download and install the client. Assume that the installation directory is **/opt/client**. Go to **/opt/client/Spark2x/spark/bin** and run the **mergetool.sh** script.

Environment variables loading

source /opt/client/bigdata_env

source /opt/client/Spark2x/component_env

Scanning function

Command: **sh mergetool.sh scan <db.table> <filesize>**

The format of *db.table* is *Database name*,*Table name*. *filesize* is the user-defined threshold of the small file size (unit: MB). The returned result is the number of files that is smaller than the threshold and the average size of data files in the table directory.

Example: **sh mergetool.sh scan default.table1 128**

Combination function

Command: **sh mergetool.sh merge <db.table> <filesize> <shuffle>**

The format of *db.table* is *Database name,Table name*. **filesize** is the user-defined average file size after file combination (unit: MB). **shuffle** is a Boolean value, and the value is **true** or **false**, which is used to configure whether to allow data to be shuffled during the merge.

Example: **sh mergetool.sh merge default.table1 128 false**

If the following information is displayed, the operation is successful:

```
SUCCESS: Merge succeeded
```

NOTE

1. Ensure that the current user is the owner of the merged table.
2. Before combination, ensure that HDFS has sufficient storage space, greater than the size of the combined table.
3. Table data must be combined separately. If a table is read during table data combination, the file may not be found temporarily. After the combination is complete, this problem is resolved. During the combination, do not write data to the corresponding tables. Otherwise, data inconsistency may occur.
4. If you encounter an error stating that the file does not exist while querying data in a partitioned table using the connected session of spark-beeline/spark-sql, run the **refresh table** *Table name* command as prompted to retry the query.
5. Configure **filesize** based on the site requirements. For example, you can set **filesize** to a value greater than the average during file merging after obtaining the average file size by file scan. Otherwise, the number of files may increase after the file merging.
6. During the file merging, data in the original tables is removed to the recycle bin. In the case of any exception occurs on the data after file merging, the data in the original tables is used to replace the damaged data. If an exception occurs during the process, restore the data in the **trash** directory by using the **mv** command in HDFS.
7. In the HDFS router federation scenario, if the target NameService of the table root path is different from that of the root path **/user**, you need to manually clear the original table files stored in the recycle bin during the second combination. Otherwise, the combination fails.
8. This tool uses the configuration of the client. Performance optimization can be performed modifying required configuration in the client configuration file.

shuffle configuration

For the combination function, you can roughly estimate the change on the number of partitions before and after the combination.

Generally, if the number of old partitions is greater than the number of new partitions, set **shuffle** to **false**. However, if the number of old partitions is much greater than that of new partitions (for example, more than 100 times), you can set **shuffle** to **true** to increase the degree of parallelism and improve the combination speed.

NOTICE

- If **shuffle** is set to **true** (repartition), the performance is improved. However, due to the particularity of the Parquet and ORC storage modes, repartition will reduce the compression ratio and the total size of the table in HDFS increases by 1.3 times.
- If **shuffle** is set to **false** (coalesce), the merged files may have some difference in size, which is close to the value of the configured **filesize**.

Log storage location

The default log storage location is **/tmp/SmallFilesLog.log4j**. To customize the log storage location, configure **log4j.appender.logfile.File** in **/opt/client /Spark2x/spark/tool/log4j.properties**.

24.8.8 Configuring Streaming Reading of Spark Driver Execution Results

Scenario

When a query statement is executed, the returned result may be large (containing more than 100,000 records). In this case, JDBCServer out of memory (OOM) may occur. Therefore, the data aggregation function is provided to avoid OOM without sacrificing the performance.

Configuration

Two data aggregation function configuration parameters are provided. The two parameters are set in the **tunning** option on the Spark JDBCServer server. After the setting is complete, restart JDBCServer.

Table 24-19 Parameter description

Parameter	Description	Default Value
spark.sql.bigdata.thriftServer.useHdfsCollect	Indicates whether to save result data to HDFS instead of the memory. Advantages: The query result is stored in HDFS. Therefore, JDBCServer OOM does not occur. Disadvantages: The query is slow. ● **true**: Result data is saved to HDFS. ● **false**: This function is disabled. **NOTICE** When **spark.sql.bigdata.thriftServer.useHdfsCollect** is set to **true**, result data is saved to HDFS. However, the job description on the native JobHistory page cannot be associated with the corresponding SQL statement. In addition, the execution ID in the spark-beeline command output is null. To solve the JDBCServer OOM problem and ensure correct information display, you are advised to set **spark.sql.userlocalFileCollect**.	false
spark.sql.uselocalFileCollect	Indicates whether to save result data to the local disk instead of memory. Advantages: In the case of small data volume, the performance loss can be ignored compared with the data storage mode using the native memory. In the case of large data volume (hundreds of millions of data records), the performance is much better than that when data is stored in the HDFS and native memory. Disadvantages: Optimization is required. In the case of large data volume, it is recommended that the JDBCServer driver memory be 10 GB and each core of the executor be allocated with 3 GB memory. ● **true**: This function is enabled. ● **false**: This function is disabled.	false

Parameter	Description	Default Value
spark.sql.collect.Hive	This parameter is valid only when **spark.sql.uselocalFileCollect** is set to **true**. It indicates whether to save the result data to a disk in direct serialization mode or in indirect serialization mode. Advantage: For queries of tables with a large number of partitions, the aggregation performance of the query results is better than that of the storage mode that query results are directly stored on the disk. Disadvantages: The disadvantages are the same as those when **spark.sql.uselocalFileCollect** is enabled. • **true**: This function is enabled. • **false**: This function is disabled.	false
spark.sql.collect.serialize	This parameter takes effect only when both **spark.sql.uselocalFileCollect** and **spark.sql.collect.Hive** are set to **true**. The function is to further improve performance. • **java**: Data is collected in Java serialization mode. • **kryo**: Data is collected in kryo serialization mode. The performance is better than that when the Java serialization mode is used.	java

NOTE

spark.sql.bigdata.thriftServer.useHdfsCollect and **spark.sql.uselocalFileCollect** cannot be set to **true** at the same time.

24.8.9 Enabling a Spark Executor to Execute Custom Code When Exiting

NOTE

This section applies only to MRS 3.2.0 or later.

Scenario

You can configure the following parameters to execute custom code when Executor exits.

Configuration Parameters

Configure the following parameters in the **spark-defaults.conf** file of the Spark client.

Parameter	Description	Default Value
spark.executor.execute.shutdown.cleaner	If this parameter is set to **true**, an executor can execute custom code when the executor exits.	false
spark.executor.execute.shutdown.cleaner.max.timeout	Timeout interval for an executor to execute custom code.	240s

24.8.10 Configuring Spark Dynamic Masking

NOTE

- This section is available for MRS 3.3.1-LTS or later version only.
- The dynamic data masking feature cannot be enabled if jobs are submitted on the console.

Scenario

Enabling Spark dynamic masking allows for the utilization of data within the masked column for computations, while keeping it concealed during the output of calculation results. The cluster's masking policy is dynamically transferred in accordance with lineage relationships, optimizing data utility while safeguarding privacy.

Constraints

- Data masking is not applicable to Hudi tables.
- Masking for non-SQL methods is not supported.
- Masking for direct HDFS read/write operations is not supported.
- Masking for complex data types like arrays, maps, and structs is not supported.
- Spark jobs are restricted to submission via spark-beeline (JDBC connection) mode.
- In instances where the masking policy transfer results in a conflict with an existing policy on the target table, the latter's policy will be overridden as **Custom: *****.
- Presently, data types such as int, char, varchar, date, decimal, float, bigint, timestamp, tinyint, smallint, double, string, and binary are amenable to data masking. Post-policy configuration for data types like int, date, decimal, float, bigint, timestamp, tinyint, smallint, and double, discrepancies may arise between the spark-beeline query outcome and anticipated results; the output will not reflect original values. To align query results with policy expectations, employing the Nullify data masking policy is advised.

- For data types not supported by the data masking policy, or when data masking transfer is implicated in the output column, the Nullify policy is the default recourse.

Procedure

1. Modify the JDBCServer instance configuration. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark**, click **Configurations**, click **All Configurations**, and choose **JDBCServer(Role)**.
   - If you plan to use Ranger authentication, add the following custom parameters in the **custom** area:

Parameter	Value
spark.dynamic.masked.enabled	true
spark.ranger.plugin.authorization.enable	true

     Modify the following parameter:

Parameter	Value
spark.ranger.plugin.masking.enable	true
spark.sql.authorization.enabled	true

   - If you plan to use Hive metadata authentication instead of Ranger authentication, add the following custom parameters in the **custom** area:

Parameter	Value
spark.ranger.plugin.use.hive.acl.enable	true
spark.dynamic.masked.enabled	true
spark.ranger.plugin.authorization.enable	false

     Modify the following parameter:

Parameter	Value
spark.ranger.plugin.masking.enable	true

NOTE

1. If you plan to use Hive metadata authentication instead of Ranger authentication and Hive policy initialization is not complete in Ranger, perform the following operations:
   - Enable the Ranger authentication function of Hive and restart Hive and Spark.
   - Enable the Ranger authentication function of Spark and restart Spark.
   - Disable the Ranger authentication function of Hive and restart Hive.
   - Disable the Ranger authentication function of Spark and restart Spark.
2. Log in to the Ranger web UI. If the Hive component exists under **Hadoop SQL**, the Hive policy has been initialized. Otherwise, the Hive policy has not been initialized.
3. If the HetuEngine component is installed in the cluster and the masking policies of the Ranger and HetuEngine spaces need to be automatically updated when the Spark dynamic masking policy is transferred, set **spark.dynamic.masked.hetu.policy.sync.update.enable** to **true**. You also need to change the Ranger user type of the built-in user Spark2x to **admin**.

2. Save the configuration and restart the Spark service.
3. Log in to the Spark client node and run the following commands:

   cd *Client installation directory*

   source bigdata_env

   source Spark/component_env

   For clusters with Kerberos authentication enabled, additionally run the following command:

   kinit *test* (Enter the password for authentication and change the password upon your first login.)
4. Run the beeline commands of Spark to submit a task and create a Spark table.

   spark-beeline

   create table *sparktest(a int, b string)*;

   insert into *sparktest* **values** *(1,"test01"), (2,"test02")*;
5. Configure a masking policy for the **sparktest** table and check whether the masking takes effect. For details, see **Adding a Ranger Access Permission Policy for Spark2x**.

   select * from sparktest;

```
0: jdbc:hive2://192-168-43-237:22550/> select * from sparktest;
INFO  : Execution ID: 9
+----+---------+
| a  |    b    |
+----+---------+
| 1  | xxxxnn  |
| 2  | xxxxnn  |
+----+---------+
2 rows selected (1.894 seconds)
```

6. Verify the transfer of the data masking policy.

   create table *sparktest02* **as select * from** *sparktest*;

   select * from *sparktest02*;


```
0: jdbc:hive2://192-168-43-237:22550/> create table sparktest02 as select * from sparktest;
INFO  : Execution ID: 11
+---------+
| Result  |
+---------+
+---------+
No rows selected (11.171 seconds)
0: jdbc:hive2://192-168-43-237:22550/> select * from sparktest02;
INFO  : Execution ID: 14
+----+---------+
| a  |    b    |
+----+---------+
| 1  | xxxxnn  |
| 2  | xxxxnn  |
+----+---------+
```

Should the information above be displayed, it indicates the dynamic masking configuration is operational. Access the Ranger masking policy management page to view the automatically generated masking policy for the **sparktest02** table.

24.8.11 Configuring Distinct Aggregation Optimization

NOTE

This section is available for MRS 3.3.1-LTS or later version only.

Scenario

In SQL statements featuring multiple **count(distinct)** aggregation functions alongside operators that induce data expansion, such as **cube** and **rollup**, enabling this feature can significantly reduce the data multiplication factor. This optimization minimizes the amount of data shuffled to disk, thereby enhancing performance. Once enabled, the **count(distinct)** operator's implementation transitions from an "expand+multi-round aggregation" to a streamlined "count_distinct" aggregation function.

Constraints

Sufficient memory has been configured for the job.

Configuring Parameters

Modify the following parameters in the *Client installation directory*/**Spark/spark/conf/spark-defaults.conf** file on the Spark client.

Parameter	Description	Default Value
spark.sql.keep.distinct.expandThreshold	This parameter determines the threshold for activating this optimization in scenarios where data expansion due to **cube** operations is significant. Setting this parameter to a positive integer, such as **1024**, triggers the optimization when the data volume expands by a factor of **1024** or more.	-1

Parameter	Description	Default Value
spark.sql.distinct.aggregator.enabled	This parameter controls the enforcement of distinct aggregation optimization. With this function enabled, the distinct aggregation is restructured without being constrained by the data expansion factor. Use this setting only when you ensure it is beneficial.	false

24.8.12 Clearing Residual Files When a Spark Job Fails to Be Configured

NOTE

This section is available for MRS 3.3.1-LTS or later version only.

Scenario

When Spark jobs fail, residual files may linger, potentially triggering disk space alarms due to their accumulation over time. It is advisable to routinely clean up these residual files.

Constraints

- To use this feature, you need to start the Spark JDBCServer service. The resident process of the JDBCServer service is used to periodically delete residual files.
- This feature also requires the configuration and modification of Spark client parameters and Spark JDBCServer server parameters.
- The following directories can be cleared:
  - **/user/***User***/.sparkStaging/**
  - **/tmp/sparkhive-scratch/***User*
- This feature supports only the scenario where Yarn is used as the resource scheduler.

Parameter Configuration

1. Modify the following parameters in the *Client installation directory***/Spark/spark/conf/spark-defaults.conf** file on the Spark client.

Parameter	Mandatory	Default Value
spark.yarn.session.to.application.clean.enabled	If this parameter is set to **true**, Spark periodically deletes residual files.	false

2. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark**, click **Configurations**, and click **All Configurations**. On the displayed page, click JDBCServer(Role) and then **Custom**. Add the following parameters in the **custom** area, and restart the JDBCServer service.

Parameter	Mandatory	Default Value
spark.yarn.session.to.application.clean.enabled	If this parameter is set to **true**, Spark periodically deletes residual files.	false
spark.clean.residual.tmp.dir.init.delay	Initial delay for clearing files, in minutes.	5
spark.clean.residual.tmp.dir.period.delay	Interval for deleting files, in minutes.	10

24.8.13 Configuring Spark to Load Third-Party JAR Packages for UDF Registration or SparkSQL Extension

NOTE

This section is available for MRS 3.5.0-LTS or later version only.

Scenarios

To enhance Spark's capabilities, custom UDFs or JAR packages are frequently used. However, to use these third-party JAR packages, you must specify the third-party class loading path before starting Spark.

Prerequisites

Custom JAR package has been uploaded to the client node. This section uses **spark-test.jar** as an example to describe how to upload the package to the **/tmp** directory on the client node.

Configuring Parameters

Step 1 Log in to the node where the client is installed as the client installation user and load environment variables.

cd *Client installation directory*

source bigdata_env

If Kerberos authentication has been enabled for the cluster (in security mode), run the following command for user authentication. If Kerberos authentication is not enabled for the cluster (in normal mode), user authentication is not required.

kinit *Component service user*

Step 2 Upload the JAR package to the HDFS, for example, **hdfs://hacluster/tmp/spark/JAR**.

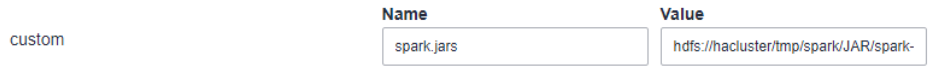
hdfs dfs -put /tmp/spark-test.jar /tmp/spark/JAR/

Step 3 Modify the following parameters in the *Client installation directory*/**Spark/spark/conf/spark-defaults.conf** file on the Spark client.

Parameter	Value
spark.jars	JAR package path, for example, **hdfs://hacluster/tmp/spark/JAR/spark-test.jar**.

Step 4 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark**, click **Configurations**, and click **All Configurations**. On the displayed page, click JDBCServer(Role) and then **Custom**. Add the following parameters in the **custom** area, and restart the JDBCServer service.

Parameter	Value
spark.jars	JAR package path, for example, **hdfs://hacluster/tmp/spark/JAR/spark-test.jar**.



Step 5 Verify that the JAR package has been loaded and the execution result does not contain "ClassNotFoundException".

----End

24.9 Spark SQL Enterprise-Class Enhancements

24.9.1 Configuring Vector-based ORC Data Reading

Scenario

ORC is a column-based storage format in the Hadoop ecosystem. It originates from Apache Hive and is used to reduce the Hadoop data storage space and accelerate the Hive query speed. Similar to Parquet, ORC is not a pure column-based storage format. In the ORC format, the entire table is split based on the row group, data in each row group is stored by column, and data is compressed as

much as possible to reduce storage space consumption. Vector-based ORC data reading significantly improves the ORC data reading performance. In Spark2.3, SparkSQL supports vector-based ORC data reading (this function is supported in earlier Hive versions). Vector-based ORC data reading improves the data reading performance by multiple times.

This feature can be enabled by using the following parameter.

- **spark.sql.orc.enableVectorizedReader**: specifies whether vector-based ORC data reading is supported. The default value is **true**.
- **spark.sql.codegen.wholeStage**: specifies whether to compile all stages of multiple operations into a Java method. The default value is **true**.
- **spark.sql.codegen.maxFields**: specifies the maximum number of fields (including nested fields) supported by all stages of codegen. The default value is **100**.
- **spark.sql.orc.impl**: specifies whether Hive or Spark SQL native is used as the SQL execution engine to read ORC data. The default value is **hive**.

Parameters

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark2x** > **Configurations**, click **All Configurations**, and search for the following parameters.

Parameter	Description	Default Value	Value Range
spark.sql.orc.enableVectorizedReader	Specifies whether vector-based ORC data reading is supported. The default value is **true**.	true	[true,false]
spark.sql.codegen.wholeStage	Specifies whether to compile all stages of multiple operations into a Java method. The default value is **true**.	true	[true,false]
spark.sql.codegen.maxFields	Specifies the maximum number of fields (including nested fields) supported by all stages of codegen. The default value is **100**.	100	Greater than 0
spark.sql.orc.impl	Specifies whether Hive or Spark SQL native is used as the SQL execution engine to read ORC data. The default value is **hive**.	hive	[hive,native]

NOTE

1. To use vector-based ORC data reading of SparkSQL, the following conditions must be met:
   - **spark.sql.orc.enableVectorizedReader** must be set to **true** (default value). Generally, the value is not changed.
   - **spark.sql.codegen.wholeStage** must be set to **true** (default value). Generally, the value is not changed.
   - The value of **spark.sql.codegen.maxFields** must be greater than or equal to the number of columns in scheme.
   - All data is of the AtomicType. Specifically, data is not null or of the UDT, array, or map type. If there is data of the preceding types, expected performance cannot be obtained.
   - **spark.sql.orc.impl** must be set to **native**. The default value is **hive**.
2. If a task is submitted using the client, modification of the following parameters takes effect only after you download the client again: **spark.sql.orc.enableVectorizedReader**, **spark.sql.codegen.wholeStage**, **spark.sql.codegen.maxFields**, and **spark.sql.orc.impl**.

24.9.2 Filtering Partitions Without Paths in a Partitioned Table

Scenarios

When you perform the ***select*** query in a Hive partitioned table, the FileNotFoundException exception is displayed if a specified partition path does not exist in HDFS. To avoid the preceding exception, configure the **spark.sql.hive.verifyPartitionPath** parameter to filter partitions without paths.

Configuration Description

Perform either of the following methods to filter partitions without paths.

- Configure the following parameters in the **spark-defaults.conf** file on the Spark driver.

  Table 24-20 Parameter description

Parameter	Description	Default Value
spark.sql.hive.verifyPartitionPath	Indicates whether to filter partitions without paths when reading Hive partitioned tables. **true**: filters partitions without paths when reading Hive partitioned tables. **false**: disables the filtering	false

- When running the spark-submit command to submit an application, configure the **--conf** parameter to filter partitions without paths.

  Example:


```
spark-submit --class org.apache.spark.examples.SparkPi  --conf spark.sql.hive.verifyPartitionPath=true
$SPARK_HOME/lib/spark-examples_*.jar
```

24.9.3 Configuring the Drop Partition Command to Support Batch Deletion

NOTE

This section applies only to MRS 3.2.0 or later.

Scenario

Currently, the **Drop Partition** command in Spark supports partition deletion using only the equal sign (=). This configuration allows multiple filter criteria to be used to delete partitions in batches, for example, **<**, **<=**, **>**, **>=**, **!>**, and **!<**.

Configuration

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark2x** > **Configurations**, click **All Configurations**, and search for the following parameters.

Parameter	Description	Default Value
spark.sql.dropPartitionsInBatch.enabled	If this parameter is set to **true**, the **Drop Partition** command supports the following filter criteria: **<**, **<=**, **>**, **>=**, **!>**, and **!<**.	true
spark.sql.dropPartitionsInBatch.limit	Indicates the maximum number of partitions that can be batch dropped.	1000

24.9.4 Configuring Dynamic Overwriting for Hive Table Partitions

Scenario

In earlier versions, when the **insert overwrite** syntax is used to overwrite partition tables, only partitions with specified expressions are matched, and partitions without specified expressions are deleted. In Spark2.3, partitions without specified expressions are automatically matched. The syntax is the same as that of the dynamic partition matching syntax of Hive.

Parameters

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark2x** > **Configurations**, click **All Configurations**, and search for the following parameters.

Parameter	Description	Default Value	Value Range
spark.sql.sources.partitionOverwrite-Mode	Specifies the mode for inserting data in partition tables by running the **insert overwrite** command, which can be **STATIC** or **DYNAMIC**. • When it is set to **STATIC**, Spark deletes all partitions based on the matching conditions. • When it is set to **DYNAMIC**, Spark matches partitions based on matching conditions and dynamically matches partitions without specified conditions.	STATIC	[STATIC,DYNAMIC]

24.9.5 Configuring Spark SQL to Enable the Adaptive Execution Feature

Scenario

The Spark SQL adaptive execution feature enables Spark SQL to optimize subsequent execution processes based on intermediate results to improve overall execution efficiency. The following features have been implemented:

1. Automatic configuration of the number of shuffle partitions

   Before the adaptive execution feature is enabled, Spark SQL specifies the number of partitions for a shuffle process by specifying the **spark.sql.shuffle.partitions** parameter. This method lacks flexibility when multiple SQL queries are performed on an application and cannot ensure optimal performance in all scenarios. After adaptive execution is enabled, Spark SQL automatically configures the number of partitions for each shuffle process, instead of using the general configuration. In this way, the proper number of partitions is automatically used during each shuffle process.

2. Dynamic adjusting of the join execution plan

   Before the adaptive execution feature is enabled, Spark SQL creates an execution plan based on the optimization results of rule-based optimization (RBO) and Cost-Based Optimization (CBO). This method ignores changes of result sets during data execution. For example, when a view created based on a large table is joined with other large tables, the execution plan cannot be adjusted to BroadcastJoin even if the result set of the view is small. After the adaptive execution feature is enabled, Spark SQL can dynamically adjust the execution plan based on the execution result of the previous stage to obtain better performance.

3. Automatic processing of data skew

   If data skew occurs during SQL statement execution, the memory overflow of an executor or slow task execution may occur. After the adaptive execution feature is enabled, Spark SQL can automatically process data skew scenarios. Multiple tasks are started for partitions where data skew occurs. Each task reads several output files obtained from the shuffle process and performs union operations on the join results of these tasks to eliminate data skew.

Parameters

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark2x** > **Configurations**, click **All Configurations**, and search for the following parameter.

Parameter	Description	Default Value
spark.sql.adaptive.enabled	Specifies whether to enable the adaptive execution function. Note: If AQE and Static Partition Pruning (DPP) are enabled at the same time, DPP takes precedence over AQE during SparkSQL task execution. As a result, AQE does not take effect.	false
spark.sql.optimizer.dynamicPartitionPruning.enabled	The switch to enable DPP.	true
spark.sql.adaptive.coalescePartitions.enabled	If this parameter is set to **true** and **spark.sql.adaptive.enabled** is set to **true**, Spark combines partitions that are consecutively random played based on the target size (specified by **spark.sql.adaptive.advisoryPartitionSizeInBytes**) to prevent too many small tasks from being executed.	true
spark.sql.adaptive.coalescePartitions.initialPartitionNum	Initial number of shuffle partitions before merge. The default value is the same as the value of **spark.sql.shuffle.partitions**. This parameter is valid only when **spark.sql.adaptive.enabled** and **spark.sql.adaptive.coalescePartitions.enabled** are set to **true**. This parameter is optional. The initial number of partitions must be a positive number.	200

Parameter	Description	Default Value
spark.sql.adaptive.coalescePartitions.minPartitionNum	Minimum number of shuffle partitions after merge. If this parameter is not set, the default degree of parallelism (DOP) of the Spark cluster is used. This parameter is valid only when **spark.sql.adaptive.enabled** and **spark.sql.adaptive.coalescePartitions.enable** are set to **true**. This parameter is optional. The initial number of partitions must be a positive number.	1
spark.sql.adaptive.shuffle.targetPostShuffleInputSize	Target size of a partition after shuffling. Spark 3.0 and later versions do not support this parameter.	64MB
spark.sql.adaptive.advisoryPartitionSizeInBytes	Size of a shuffle partition (unit: byte) during adaptive optimization (**spark.sql.adaptive.enabled** is set to **true**). This parameter takes effect when Spark aggregates small shuffle partitions or splits shuffle partitions where skew occurs.	64MB
spark.sql.adaptive.fetchShuffleBlocksInBatch	Whether to obtain consecutive shuffle blocks in batches. For the same map job, reading consecutive shuffle blocks in batches can reduce I/Os and improve performance, instead of reading blocks one by one. Note that multiple consecutive blocks exist in a single read request only when **spark.sql.adaptive.enabled** and **spark.sql.adaptive.coalescePartitions.enabled** are set to **true**. This feature also relies on a relocatable serializer that uses cascading to support the codec and the latest version of the shuffle extraction protocol.	true
spark.sql.adaptive.localShuffleReader.enabled	If the value of this parameter is **true** and the value of **spark.sql.adaptive.enabled** is **true**, Spark attempts to use the local shuffle reader to read shuffle data when shuffling of partitions is not required, for example, after sort-merge join is converted to broadcast-hash join.	true

Parameter	Description	Default Value
spark.sql.adaptive.skewJoin.enabled	Specifies whether to enable the function of automatic processing of the data skew in join operations. The function is enabled when this parameter is set to **true** and **spark.sql.adaptive.enabled** is set to **true**.	true
spark.sql.adaptive.skewJoin.skewedPartitionFactor	This parameter is a multiplier used to determine whether a partition is a data skew partition. If the data size of a partition exceeds the value of this parameter multiplied by the median of the all partition sizes except this partition and exceeds the value of **spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes**, this partition is considered as a data skew partition.	5
spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes	If the partition size (unit: byte) is greater than the threshold as well as the product of the **spark.sql.adaptive.skewJoin.skewedPartitionFactor** value and the median partition size, skew occurs in the partition. Ideally, the value of this parameter should be greater than that of **spark.sql.adaptive.advisoryPartitionSizeInBytes.**.	256MB
spark.sql.adaptive.nonEmptyPartitionRatioForBroadcastJoin	If the ratio of non-null partitions is less than the value of this parameter when two tables are joined, broadcast hash join cannot be properly performed regardless of the partition size. This parameter is valid only when **spark.sql.adaptive.enabled** is set to **true**.	0.2

24.9.6 Using Spark SQL Statements Without Aggregate Functions for Correlated Subqueries

NOTE

This section is available for MRS 3.3.1-LTS or later version only.

Scenario

If you are using open-source Spark SQL, the aggregate function must be used for correlated subqueries. Otherwise, "Error in query: Correlated scalar subqueries must be aggregated" will be reported. MRS allows you to perform correlated subqueries without using aggregate functions.

Constraints

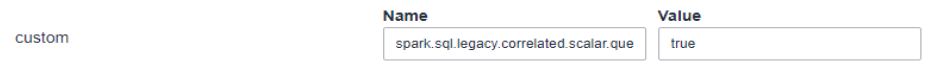
- SQL statements similar to **select id, (select group_name from emp2 b where a.group_id=b.group_id) as banji from emp1 a** is supported.
- SQL statements similar to **select id, (select distinct group_name from emp2 b where a.group_id=b.group_id) as banji from emp1 a** is supported.

Parameter Configuration

- Spark SQL scenario: Modify the following parameters in the *Client installation directory*/**Spark/spark/conf/spark-defaults.conf** file on the Spark client.

Parameter	Mandatory	Default Value
spark.sql.legacy.correlated.scalar.query.enabled	If this parameter is set to **true**, Spark supports correlated subqueries without aggregate functions.	false

- In the spark-beeline scenario, configure JDBCServer custom parameters.
  a. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark**, choose **Configurations** > **All Configurations**, and choose **JDBCServer (Role)** > **Custom**. Add the **spark.sql.legacy.correlated.scalar.query.enabled** parameter in the **Custom** area and set it to **true**.



  b. Click **Save**. Click **Instances**, select all JDBCServer instances, choose **More** > **Restart Instance**, and operate as prompted.

NOTE

If a correlated subquery uses multiple match predicates, an exception occurs.

24.10 Spark Streaming Enterprise-Class Enhancements

24.10.1 Configuring the LIFO Function When Spark Streaming Interconnects with Kafka

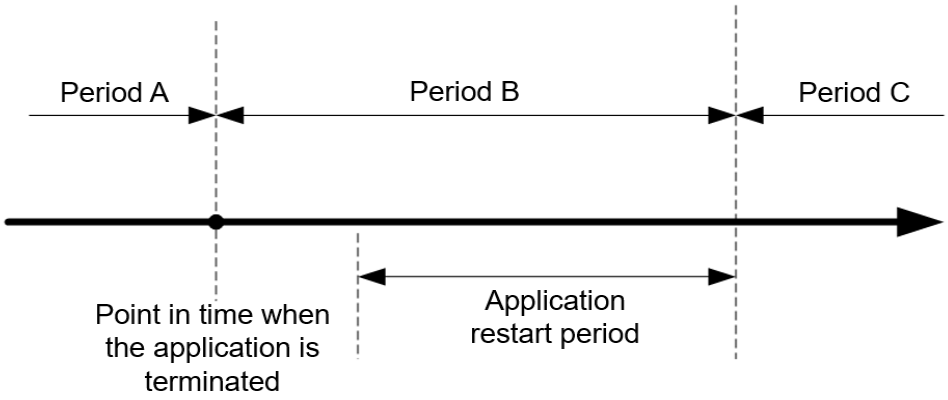
Scenario

If the Spark Streaming application is connected to Kafka, after the Spark Streaming application is terminated abnormally and restarted from the

checkpoint, the system preferentially processes the tasks that are not completed before the application is terminated (Period A) and the tasks generated based on data that enters Kafka during the period (Period B) from the application termination to the restart. Then the application processes the tasks generated based on data that enters Kafka after the application is restarted (Period C). For data that enters Kafka in period B, Spark generates a corresponding number of tasks based on the end time (**batch** time). The first task reads all data, but other tasks may not read data. As a result, the task processing pressure is uneven.

If the tasks in Period A and Period B are processed slowly, the processing of tasks in period C is affected. To cope with the preceding scenario, Spark provides the last-in first-out (LIFO) function for Kafka.

Figure 24-5 Time axis for restarting the Spark Streaming application



After this function is enabled, Spark preferentially schedules tasks in Period C. If there are multiple tasks in Period C, Spark schedules and executes the tasks in the sequence of task generation. Then Spark executes the tasks in Periods A and B. For data that enters Kafka in Period B, Spark generates tasks based on the end time and evenly distributes all data that enters Kafka in this period to each task to avoid uneven task processing pressure.

Constraints:

- This function applies only to the direct mode of Spark Streaming, and the execution result does not depend on the processing result of the previous batch (that is, stateless operation, for example, **updatestatebykey**). Multiple data input streams must be comparatively independent from each other. Otherwise, the result may change after the data is divided.
- The Kafka LIFO function can be enabled only when the application is connected to the Kafka input source.
- If both Kafka LIFO and flow control functions are enabled when the application is submitted, the flow control function is not enabled for the data that enters Kafka in Period B to ensure that the task scheduling priority for reading the data is the lowest. Flow control is enabled for the tasks in Period C after the application is restarted.

Configuration

Configure the following parameters in the **spark-defaults.conf** file on the Spark driver.

Table 24-21 Parameter description

Parameter	Description	Default Value
spark.streaming.kafka.direct.lifo	Specifies whether to enable the LIFO function of Kafka.	false
spark.streaming.kafka010.inputstream.class	Obtains the decoupled class on FusionInsight.	org.apache.spark.streaming.kafka010.xxDirectKafkaInputDStream

24.10.2 Configuring Reliability of Interconnection Between Spark Streaming and Kafka

Scenario

When the Spark Streaming application is connected to Kafka and the application is restarted, the application reads data from Kafka based on the last read topic offset and the latest offset of the current topic.

If the leader of a Kafka topic fails and the offset of the Kafka leader is greatly different from that of the Kafka follower, the Kafka follower and leader are switched over after the Kafka service is restarted. As a result, the offset of the topic decreases after the Kafka service is restarted.

- If the Spark Streaming application keeps running, the start position for reading Kafka data is greater than the end position because the offset of the topic in Kafka decreases. As a result, the application cannot read data from Kafka and reports an error.
- Before restarting the Kafka service, stop the Spark Streaming application. After the Kafka service is restarted, restart the Spark Streaming application to restore the application from the checkpoint. In this case, the Spark Streaming application records the offset position read before the termination and uses the position as the reference to read subsequent data. The Kafka offset decreases (for example, from 100,000 to 10,000). Spark Streaming consumes data only after the offset of the Kafka leader increases to 100,000. As a result, the newly sent data whose offset is between 10,000 and 100,000 is lost.

To resolve the preceding problem, you can configure reliability for Kafka connected to Spark Streaming. After the reliability function of connected Kafka is enabled:

- When the Spark Streaming application is running and the offset of a topic in Kafka decreases, the start position for reading Kafka data will be set to the latest offset of the topic in Kafka, and the application will continue to read subsequent data.

If a task has been generated but not yet scheduled, and the read Kafka offset is higher than the latest offset of the topic in Kafka, the task will fail to execute.

NOTE

If a large number of tasks fail, the Executor is added to the blacklist. As a result, subsequent tasks cannot be deployed and run. If this happens, you can set **spark.blacklist.enabled** to disable the blacklist function. The blacklist function is enabled by default.

- If the offset of a topic in Kafka decreases, the Spark Streaming application restarts to restore the unfinished tasks. If the read Kafka offset range is greater than the latest offset of the topic in Kafka, the task is directly discarded.

NOTE

If the state function is used in the Spark Streaming application, do not enable the reliability function of connected Kafka.

Configuration

Configure the following parameter in the **spark-defaults.conf** file of the Spark client.

Table 24-22 Parameter description

Parameter	Description	Default Value
spark.streaming.Kafka.reliability	Indicates whether to enable the reliability function for Kafka connected to Spark Streaming. • **true**: The reliability function is enabled. • **false**: The reliability function is disabled.	false

24.10.3 Configuring Structured Streaming to Use RocksDB for State Store

NOTE

This section applies only to MRS 3.3.0 or later.

Scenarios

If a large amount of state information is stored in the default HDFS BackedStateStore and JVM GC takes a long time, you can use the following method to select RocksDB as the state backend.

Parameters

Configure the following parameters in the **spark-defaults.conf** file of the Spark client.

Parameter	Description	Default Value
spark.sql.streaming.stateStore.providerClass	Class that manages state data for stateful stream queries. This class must be a subclass of StateStoreProvider and must have a zero argument constructor. Set this parameter to **org.apache.spark.sql.execution.streaming.state.RocksDBStateStoreProvider** to select RocksDB as the state backend.	org.apache.spark.sql.execution.streaming.state.HDFSBackedStateStoreProvider

24.11 Spark Core Performance Tuning

24.11.1 Spark Core Data Serialization

Scenario

Spark supports the following types of serialization:

- JavaSerializer
- KryoSerializer

Data serialization greatly affects the Spark application performance. In specific data format, KryoSerializer offers 10 times higher performance than JavaSerializer. For data of int type, performance optimization can be ignored.

KryoSerializer depends on Chill of Twitter. Not all Java Serializable objects support KryoSerializer. Therefore, a class must be manually registered.

Serialization involves task serialization and data serialization. Only JavaSerializer can be used for Spark task serialization. JavaSerializer and KryoSerializer can be used for data serialization.

Procedure

When the Spark application is running, a large volume of data needs to be serialized during the shuffle and RDD cache procedures. By default, JavaSerializer is used. You can also configure KryoSerializer as the data serializer to improve serialization performance.

When developing an application, add the following code to enable KryoSerializer as data serializer:

- Implement the class register and manually register classes.

```
package com.etl.common;

import com.esotericsoftware.kryo.Kryo;
import org.apache.spark.serializer.KryoRegistrator;
```

```
public class DemoRegistrator implements KryoRegistrator
{
   @Override
   public void registerClasses(Kryo kryo)
   {
      //The following is an example class. Please register a custom class.
      kryo.register(AggrateKey.class);
      kryo.register(AggrateValue.class);
   }
}
```

You can configure **spark.kryo.registrationRequired** on a Spark client to determine whether registration with KryoSerializer is required.

If the parameter is set to **true**, an exception is thrown if a project has classes that are not serialized. If the parameter is set to **false** (default value), KryoSerializer automatically writes unregistered classes to the corresponding objects. This operation affects system performance. If the parameter is set to **true**, you must manually register classes. The system does not write classes that are not serialized but throws exceptions. System performance is not affected.

- Configure KryoSerializer as the data serializer and class register.

```
val conf = new SparkConf()
conf.set("spark.serializer", "org.apache.spark.serializer.KryoSerializer")
.set("spark.kryo.registrator", "com.etl.common.DemoRegistrator")
```

24.11.2 Spark Core Memory Tuning

Scenario

Spark is an in-memory computing frame. If the memory is insufficient during computing, the Spark execution efficiency will be adversely affected. You can determine whether the memory becomes a performance bottleneck by monitoring garbage collection (GC) and evaluating the resilient distributed dataset (RDD) size in the memory, and take performance optimization measures.

To monitor GC of node processes, add the **-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps** parameter to the **spark.driver.extraJavaOptions** and **spark.executor.extraJavaOptions** configuration items in the **conf/spark-default.conf** configuration file of the client.

If "Full GC" is frequently reported, GC needs to be optimized. Cache the RDD and query the RDD size in the log. If a large value is found, change the RDD storage level.

Procedure

- To optimize GC, adjust the size and ratio of the old generation and young generation. In the **conf/spark-default.conf** configuration file of the client, add the **-XX:NewRatio** parameter to the **spark.driver.extraJavaOptions** and **spark.executor.extraJavaOptions** configuration items. For example, if you add **-XX:NewRatio=2**, the young generation accounts for 1/3 of the heap space, and the old generation accounts for 2/3.
- Optimize the RDD data structure when developing Spark applications.
  - Use primitive arrays to replace fastutil arrays.
  - Avoid nested structure.

  - Avoid using String in keys.
- Serialize RDDs when developing Spark applications.

  By default, data is not serialized when RDDs are cached. You can set the storage level to serialize the RDDs and minimize memory usage. The following is an example.

  ```
  testRDD.persist(StorageLevel.MEMORY_ONLY_SER)
  ```

24.11.3 Setting Spark Core DOP

Scenario

A degree of parallelism (DOP) specifies the number of tasks to be executed concurrently. It determines the number of data blocks after the shuffle operation. Configuring the DOP will optimize the number of tasks, data volume of each task, and the host processing capability.

Query the CPU and memory usage. If data and tasks are not evenly distributed among nodes, increase the DOP for even distribution. Generally, set the DOP to two or three times that of the total CPUs in the cluster.

Procedure

You can use any of the following methods to set the DOP and adjust the DOP parameters according to the actual memory, CPU, data, and application logic:

- Set the DOP parameters in the function of shuffle operations. This method has the highest priority.
  ```
  testRDD.groupByKey(24)
  ```
- Set the **spark.default.parallelism** parameter in the code. This method has the second highest preference.
  ```
  val conf = new SparkConf()
  conf.set("spark.default.parallelism", 24)
  ```
- Set the **spark.default.parallelism** parameter in the **$SPARK_HOME/conf/spark-defaults.conf** file. This method has the lowest preference.
  ```
  spark.default.parallelism    24
  ```

24.11.4 Configuring Spark Core Broadcasting Variables

Scenario

Broadcasting datasets to each node allows for local access during Spark tasks, eliminating the need for data serialization to be scheduled to tasks each time data sets are required. This not only saves time but also prevents tasks from becoming larger.

1. To ensure that a dataset is available to every slice of a task, it is recommended to broadcast the dataset to each node.
2. To avoid the shuffle operation and simplify the join process when working with small and large tables, it is best to broadcast the small tables to each node.

Procedure

When developing an application, add the following code to broadcast the testArr data to each node:

```
def main(args: Array[String]) {
  ...
  val testArr: Array[Long] = new Array[Long](200)
  val testBroadcast: Broadcast[Array[Long]] = sc.broadcast(testArr)
  val resultRdd: RDD[Long] = inpputRdd.map(input => handleData(testBroadcast, input))
  ...
}

def handleData(broadcast: Broadcast[Array[Long]], input: String) {
  val value = broadcast.value
  ...
}
```

24.11.5 Configuring Heap Memory Parameters for Spark Executor

Scenario

When the executor off-heap memory is too small, or processes with higher priority preempt resources, the physical memory usage will exceed the maximal value. To prevent the physical memory usage from exceeding, set the following parameter.

Configuration

Navigation path for setting parameters:

When submitting an application, set the following parameter using **--conf** or adjust the parameter in the **spark-defaults.conf** configuration file on the client.

Table 24-23 Parameter description

Parameter	Description	Default Value
spark.executor.memoryOverhead	Indicates the off-heap memory of each executor, in MB. Increasing the value of this parameter prevents the physical memory usage from exceeding the maximal value. The value is calculated based on max(384, Executor – Memory x 0.1). The minimal value is 384.	1024

24.11.6 Using the External Shuffle Service to Improve Spark Core Performance

Scenario

When the Spark system runs applications that contain a shuffle process, an executor process also writes shuffle data and provides shuffle data for other

executors in addition to running tasks. If the executor is heavily loaded and GC is triggered, the executor cannot provide shuffle data for other executors, affecting task running.

The external shuffle service is an auxiliary service in NodeManager. It captures shuffle data to reduce the load on executors. If GC occurs on an executor, tasks on other executors are not affected.

Procedure

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **Spark2x** and click **Configurations**. Select **All Configurations**.

Step 3 Choose **SparkResource2x** > **Default** and modify the following parameters.

Table 24-24 Parameter list

Parameter	Default Value	Changed To
spark.shuffle.service.enabled	false	true

Step 4 Restart the Spark2x service for the configuration to take effect.

NOTE

To use the External Shuffle Service function on the Spark2x client, you need to download and install the Spark2x client again.

----End

24.11.7 Configuring Spark Dynamic Resource Scheduling in YARN Mode

Scenario

Resources are a key factor that affects Spark execution efficiency. Allocating multiple executors to a long-running service, such as the JDBCServer, without tasks can result in improper scheduling and wasted resources if there are insufficient resources available for other applications.

Dynamic resource scheduling can add or remove executors of applications in real time based on the task load. In this way, resources are dynamically scheduled to applications.

Procedure

Step 1 Configure the external shuffle service.

Step 2 Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark2x**, click **Configurations**, and click **All Configurations**. Enter the

spark.dynamicAllocation.enabled parameter name in the search box and set it to **true** to enable dynamic resource scheduling.

----End

Table 24-25 lists some optional configuration items.

Table 24-25 Parameters for dynamic resource scheduling

Configuration Item	Description	Default Value
spark.dynamicAllocation.minExecutors	Indicates the minimum number of executors.	0
spark.dynamicAllocation.initialExecutors	Indicates the number of initial executors.	0
spark.dynamicAllocation.maxExecutors	Indicates the maximum number of executors.	2048
spark.dynamicAllocation.schedulerBacklog-Timeout	Indicates the first timeout period for scheduling.	1s
spark.dynamicAllocation.sustainedSchedu-lerBacklogTimeout	Indicates the second and later timeout interval for scheduling.	1s
spark.dynamicAllocation.executorIdleTime-out	Indicates the idle timeout interval for common executors.	60s
spark.dynamicAllocation.cachedExecutorI-dleTimeout	Indicates the idle timeout interval for executors with cached blocks.	• JDBCServer2x: 2147483647s • IndexServer2x: 2147483647s • SparkResource2x: 120

NOTE

The external shuffle service must be configured before using the dynamic resource scheduling function.

24.11.8 Adjusting Spark Core Process Parameters

Scenario

There are three processes in Spark on Yarn mode: driver, ApplicationMaster, and executor. The Driver and Executor handle the scheduling and running of the task. The ApplicationMaster handles the start and stop of the container.

Therefore, the configuration of the driver and executor is very important to run the Spark application. You can optimize the performance of the Spark cluster according to the following procedure.

Procedure

Step 1 Configure the driver memory.

The driver schedules tasks and communicates with the executor and the ApplicationMaster. Add driver memory when the number and parallelism level of the tasks increases.

You can configure the driver memory based on the number of the tasks.

- Set **spark.driver.memory** in **spark-defaults.conf** to a proper value.
- Add the **--driver-memory MEM** parameter to configure the memory when using the **spark-submit** command.

Step 2 Configure the number of the executors.

One core in an executor can run one task at the same time. Therefore, more tasks can be processed at the same time if you increase the number of the executors. You can add the number of the executors to increase the efficiency if resources are sufficient.

- Set **spark.executor.instance** in **spark-defaults.conf** or **SPARK_EXECUTOR_INSTANCES** in **spark-env.sh** to a proper value.
- Add the **--num-executors NUM** parameter to configure the number of the executors when using the **spark-submit** command.

Step 3 Configure the number of the executor cores.

Multiple cores in an executor can run multiple tasks at the same time, which increases the task concurrency. However, because all cores share the memory of an executor, you need to balance the memory and the number of cores.

- Set **spark.executor.cores** in **spark-defaults.conf** or **SPARK_EXECUTOR_CORES** in **spark-env.sh** to a proper value.
- When you run the **spark-submit** command, add the **--executor-cores NUM** parameter to set the number of executor cores.

Step 4 Configure the executor memory.

The executor memory is used for task execution and communication. You can increase the memory for a big task that needs more resources, and reduce the memory to increase the concurrency level for a small task that runs fast.

- Set **spark.executor.memory** in **spark-defaults.conf** or **SPARK_EXECUTOR_MEMORY** in **spark-env.sh** to a proper value.
- When you run the **spark-submit** command, add the **--executor-memory MEM** parameter to set the memory.

----End

Example

- During the **spark wordcount** calculation, the amount of data is 1.6 TB and the number of the executors is 250.

  The execution fails under the default configuration, and the **Futures timed out** and **OOM** errors occur.

  However each task of wordcount is small and runs fast, the amount of the data is big and the tasks are too many. Therefore the objects on the driver end become huge when there are many tasks. Besides the fact that the executor communicates with the driver once each task is finished, the problem of disconnection between processes caused by insufficient memory occurs.

  The application runs successfully when the memory of the Driver is set to 4 GB.
- Many errors still occurred in the default configuration when running TPC-DS test on JDBCServer, such as "Executor Lost". When there is 30 GB of driver memory, 2 executor cores, 125 executors, and 6 GB of executor memory, all tasks can be successfully executed.

24.11.9 Spark DAG Design Specifications

Scenario

Optimal program structure helps increase execution efficiency. During application programming, avoid shuffle operations and combine narrow-dependency operations.

Procedure

This topic describes how to design the DAG using the following example:

- **Data format**: Time when a vehicle passes a toll station, license plate number, toll station number, and more
- **Logic**: Two vehicles are determined to be traveling together if the following conditions are met:
  - Both vehicles pass the same toll stations in the same sequence.
  - The difference between the time that the vehicles pass the same toll station is smaller than a specified value.

There are two implementation ways for this example. **Figure 24-6** shows the logic of implementation 1 and **Figure 24-7** shows logic of implementation 2.

Figure 24-6 Implementation logic 1

Logic description:

1. Collect information about the toll stations passed by each vehicle based on the vehicle license plate number and sort the toll stations.

   The following data is obtained: vehicle license plate number 1, [(time, toll station 3), (time, toll station 2), (time, toll station 4), (time, toll station 5)]

2. Determine the sequence in which the vehicle passed through.

   (toll station 3, (vehicle license plate number 1, time, 1st toll station))

   (toll station 2, (vehicle license plate number 1, time, 2nd toll station))

   (toll station 4, (vehicle license plate number 1, time, 3rd toll station))

   (toll station 5, (vehicle license plate number 1, time, 4th toll station))

3. Aggregate data by toll station.

   toll station 1, [(vehicle license plate number 1, time, 1st toll station), (vehicle license plate number 2, time, 5th toll station), (vehicle license plate number 3, time, 2nd toll station)]

4. Determine whether the time difference that two vehicles passed through the same toll station is below the specified value. If yes, fetch information about the two vehicles.

   (vehicle license plate number 1, vehicle license plate number 2),(1st toll station, 5th toll station)

   (vehicle license plate number 1, vehicle license plate number 3),(1st toll station, 2nd toll station)

5. Aggregate data based on the vehicle license plate numbers that passed through the same toll stations.

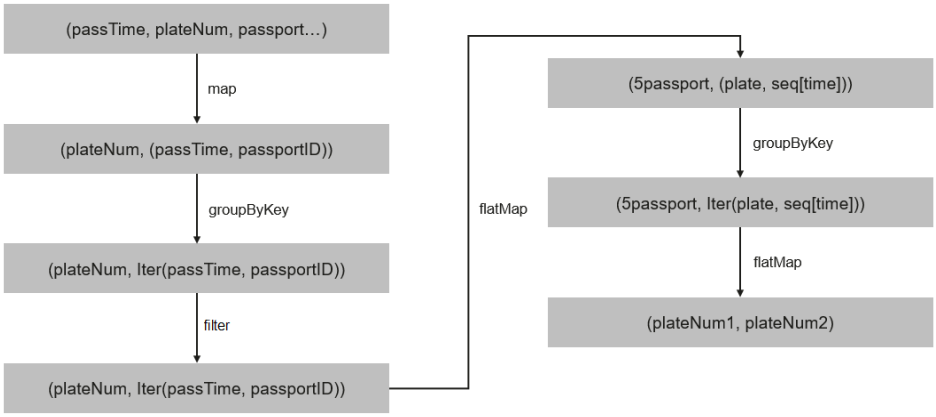
   (vehicle license plate number 1, vehicle license plate number 2), [(1st toll station, 5th toll station), (2nd toll station, 6th toll station), (1st toll station, 7th toll station), (3rd toll station, 8th toll station)]

6. If the two vehicles pass through the same toll stations in sequence, for example, toll stations 3, 4, 5 are the first, second, and third toll station passed by vehicle 1 and the 6th, 7th, and 8th toll station passed by vehicle 2, and the number of toll stations meets the specified requirements, the two vehicles are determined to be traveling together.

The logic of implementation 1 has the following disadvantages:

- The logic is complex.
- Too many shuffle operations affect performance.

Figure 24-7 Implementation logic 2



Logic description:

1. Collect information about the toll stations passed by each vehicle based on the vehicle license plate number and sort the toll stations.

   The following data is obtained: vehicle license plate number 1, [(time, toll station 3), (time, toll station 2), (time, toll station 4), (time, toll station 5)]

2. Based on the number of toll stations (the number is 3 in this example) that must be passed by these vehicles, divide the toll station sequence as follows:

   toll station 3 > toll station 2 > toll station 4, (vehicle license plate number 1, [time passing through toll station 3, time passing through toll station 2, time passing through toll station 4])

   toll station 2 > toll station 4 > toll station 5, (vehicle license plate number 1, [time passing through toll station 2, time passing through toll station 4, time passing through toll station 5])

3. Aggregate information about vehicles that pass the same toll stations in the same sequence.

   toll station 3 > toll station 2 > toll station 4, [(vehicle license plate number 1, [time passing through toll station 3, time passing through toll station 2, time passing through toll station 4]), (vehicle license plate number 2, [time passing through toll station 3, time passing through toll station 2, time passing through toll station 4]), (vehicle license plate number 3, [time passing through toll station 3, time passing through toll station 2, time passing through toll station 4])]

4. Determine whether the time difference that these vehicles passed through the same toll station is below the specified value. If yes, the vehicles are determined to be traveling together.

The logic of implementation 2 has the following advantages:

- The logic is simplified.
- One **groupByKey** is reduced, that is, one less shuffle operation is performed. It helps improve performance.

24.11.10 Experience Summary

Using mapPartitions to Calculate Data by Partition

If the overhead of each record is significant, use the following example code.

```
rdd.map{x=>conn=getDBConn;conn.write(x.toString);conn.close}
```

You can then use **mapPartitions** to calculate data by partition.

```
rdd.mapPartitions(records => conn.getDBConn;for(item <- records)
write(item.toString); conn.close)
```

The **mapPartitions** function is a versatile tool for manipulating data. For example, when working with large data sets and needing to find the top *N* values, mapPartitions can calculate the top *N* values within each partition. If *N* is a small value, the top *N* values from all partitions can then be sorted. Compared with calculating top *N* with full data, this method has a higher efficiency.

Using coalesce to Adjust the Number of Slices

Use coalesce to adjust the number of slices. There are two **coalesce** functions:

```
coalesce(numPartitions: Int, shuffle: Boolean = false)
```

When **shuffle** is set to **true**, the function is the same as repartition(numPartitions:Int). Partitions are recreated using the shuffle. When **shuffle** is set to **false**, partitions of the parent resilient distributed datasets (RDD) are calculated in the same task. In this case, if the value of **numPartitions** is greater than the number of sections of the parent RDD, partitions will not be recreated.

The following scenario is encountered, you can choose the coalesce operator:

- If the previous operation involves a large number of filters, use coalesce to minimize the number of zero-loaded tasks. In coalesce(numPartitions, false), the value of **numPartitions** is less than the number of slices of the parent RDD.
- Use **coalesce** when the number of slices entered is too large to execute.
- Use **coalesce** when the programs are suspended in the shuffle operation because of a large number of tasks or limited Linux resources. In this case, use coalesce (numPartitions, true) to recreate partitions.

Configuring localDir

During the shuffle procedure of Spark, data needs to be written into local disks. The performance bottleneck of Spark is shuffle, and the bottleneck of shuffle is the I/O. To improve the I/O performance, you can configure multiple disks to implement concurrent data writing. If multiple disks are mounted to a node, configure a Spark localDir for each disk. This can effectively distribute shuffle files in multiple locations, improving disk I/O efficiency. The performance cannot be improved if a disk is configured with multiple directories.

Using the Collect Operation for Small Data

The collect operation does not apply to a large volume of data.

When the collect operation is performed, the executor data is sent to the driver. If the driver does not have sufficient memory, **OutOfMemory** occurs on the driver. Therefore, if the data volume is unknown, perform the **saveAsTextFile** operation to write data into HDFS. If the data volume is known and the driver has sufficient memory, perform the collect operation.

Using reduceByKey

The reduceByKey operator implements local aggregation on the Map side, which offers a smooth shuffle procedure. The groupByKey operator, however, does not perform aggregation on the Map side. Therefore, use reduceByKey if possible to avoid implementation modes like groupByKey().map(x=>(x._1,x._2.size)).

Broadcasting Map Instead of Arrays

If a table query is required for each record of data that is broadcast from the driver side, broadcast the data in the set/map structure instead of Iterator. The query speed of the set/map structure is approximately O(1), while that of Iterator is O(n).

Avoiding Data Skew

If data skew occurs (certain data volume is extremely large), the execution time of tasks is inconsistent even though no GC is performed.

- Redefine keys. Use keys of smaller granularity to optimize the task size.
- Modify the DOP.

Optimizing the Data Structure

- Store data by column. In this way, only the required columns are scanned when data is read.
- When using Hash Shuffle, set **spark.shuffle.consolidateFiles** to **true** to combine intermediate files of shuffle, minimize the number of shuffle files and file I/O operations, and improve performance. The number of final files is the number of reduce tasks.

24.12 Spark SQL Performance Tuning

24.12.1 Optimizing the Spark SQL Join Operation

Scenario

When two tables are joined in Spark SQL, the broadcast function (see section "Using Broadcast Variables") can be used to broadcast tables to each node. This minimizes shuffle operations and improves task execution efficiency.

NOTE

The join operation refers to the inner join operation only.

Procedure

The following describes how to optimize the join operation in Spark SQL. Assume that both tables A and B have the **name** column. Join tables A and B as follows:

1. Estimate the table sizes.

   Estimate the table size based on the size of data loaded each time.

   You can also check the table size in the directory of the Hive database. In the **hive-site.xml** configuration file of Spark, view the Hive database directory, which is **/user/hive/warehouse** by default. The default Hive database directory for multi-instance Spark is **/user/*hive*/warehouse**, for example, **/user/hive1/warehouse**.

   ```
   <property>
     <name>hive.metastore.warehouse.dir</name>
    <value>${test.warehouse.dir}</value>
    <description></description>
   </property>
   ```

   Run the **hadoop** command to check the size of the table. For example, run the following command to view the size of table **A**:

   ```
   hadoop fs -du -s -h ${test.warehouse.dir}/a
   ```

   NOTE

   To perform the broadcast operation, ensure that at least one table is not empty.

2. Configure a threshold for automatic broadcast.

   The threshold for triggering broadcast for a table is 10485760 (that is, 10 MB) in Spark. If either of the table sizes is smaller than 10 MB, skip this step.

   Table 24-26 lists configuration parameters of the threshold for automatic broadcasting.

   Table 24-26 Parameter description

Parameter	Default Value	Description
spark.sql.autoBroadcastJoinThreshold	10485760	Indicates the maximum value for the broadcast configuration when two tables are joined. • When the size of a field in a table involved in an SQL statement is less than the value of this parameter, the system broadcasts the SQL statement. • If the value is set to **-1**, broadcast is not performed.

   Methods for configuring the threshold for automatic broadcasting:

   – Set **spark.sql.autoBroadcastJoinThreshold** in the **spark-defaults.conf** configuration file of Spark.

     ```
     spark.sql.autoBroadcastJoinThreshold = <size>
     ```


  - Run the Hive command to set the threshold. Before joining the tables, run the following command:
    ```
    SET spark.sql.autoBroadcastJoinThreshold=<size>;
    ```

3. Join the tables.
   - The size of each table is smaller than the threshold.
     - If the size of table A is smaller than that of table B, run the following command:
       ```
       SELECT A.name FROM B JOIN A ON A.name = B.name;
       ```
     - If the size of table B is smaller than that of table A, run the following command:
       ```
       SELECT A.name FROM A JOIN B ON A.name = B.name;
       ```
   - One table size is smaller than the threshold, while the other table size is greater than the threshold.

     Broadcast the smaller table.
   - The size of each table is greater than the threshold.

     Compare the size of the field involved in the query with the threshold.
     - Data in a table can be broadcast if the values of its fields are smaller than the threshold.
     - If the values of the fields in both tables exceed the threshold, it is recommended not to broadcast either of the tables.
4. (Optional) In the following scenarios, you need to run the Analyze command (***ANALYZE TABLE tableName COMPUTE STATISTICS noscan;***) to update metadata before performing the broadcast operation:
   - The table to be broadcasted is a newly created partitioned table and the file type is non-Parquet.
   - The table to be broadcasted is a newly updated partitioned table.

Reference

A task is ended if a timeout occurs during the execution of the to-be-broadcasted table.

By default, BroadCastJoin allows only 5 minutes for the to-be-broadcasted table calculation. If the time is exceeded, a timeout will occur. However, the broadcast task of the to-be-broadcasted table calculation is still being executed, resulting in resource waste.

The following methods can be used to address this issue:

- Modify the value of **spark.sql.broadcastTimeout** to increase the timeout duration.
- Reduce the value of **spark.sql.autoBroadcastJoinThreshold** to disable the optimization of BroadCastJoin.

24.12.2 Improving Spark SQL Calculation Performance Under Data Skew

Scenario

When multiple tables are joined in Spark SQL, skew occurs in join keys and the data volume in some Hash buckets is much higher than that in other buckets. As a result, some tasks are overloaded and run slowly while other tasks are light and run fast. Other tasks with a small amount of data are quickly completed, which frees many CPUs and results in a waste of CPU resources.

If the automatic data skew function is enabled, data that exceeds the bucketing threshold is bucketed. Multiple tasks proceed data in one bucket. Therefore, CUP usage is enhanced and the system performance is improved.

NOTE

Data that has no skew is bucketed and run in the original way.

Restrictions:

- Only the join between two tables is supported.
- FULL OUTER JOIN data does not support data skew.

  For example, the following SQL statement indicates that the skew of table **a** or table **b** cannot trigger the optimization.

  select aid FROM a FULL OUTER JOIN b ON aid=bid;
- LEFT OUTER JOIN data does not support the data skew of the right table.

  For example, the following SQL statement indicates that the skew of table **b** cannot trigger the optimization.

  select aid FROM a LEFT OUTER JOIN b ON aid=bid;
- RIGHT OUTER JOIN does not support the data skew of the left table.

  For example, the following SQL statement indicates that the skew of table **a** cannot trigger the optimization.

  select aid FROM a RIGHT OUTER JOIN b ON aid=bid;

Configuration Description

Add the following parameters in the following table to the **spark-defaults.conf** configuration file on the Spark driver.

Table 24-27 Parameter description

Parameter	Description	Default Value
spark.sql.adaptive.enabled	The switch to enable the adaptive execution feature. Note: If AQE and Static Partition Pruning (DPP) are enabled at the same time, DPP takes precedence over AQE during SparkSQL task execution. As a result, AQE does not take effect. The DPP in the cluster is enabled by default. Therefore, you need to disable it when enabling the AQE.	false
spark.sql.optimizer.dynamicPartitionPruning.enabled	The switch to enable DPP.	true
spark.sql.adaptive.skewJoin.enabled	Specifies whether to enable the function of automatic processing of the data skew in join operations. The function is enabled when this parameter is set to **true** and **spark.sql.adaptive.enabled** is set to **true**.	true
spark.sql.adaptive.skewJoin.skewedPartitionFactor	This parameter is a multiplier used to determine whether a partition is a data skew partition. If the data size of a partition exceeds the value of this parameter multiplied by the median of the all partition sizes except this partition and exceeds the value of **spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes**, this partition is considered as a data skew partition.	5
spark.sql.adaptive.skewjoin.skewedPartitionThresholdInBytes	If the partition size (unit: byte) is greater than the threshold as well as the product of the **spark.sql.adaptive.skewJoin.skewedPartitionFactor** value and the median partition size, skew occurs in the partition. Ideally, the value of this parameter should be greater than that of **spark.sql.adaptive.advisoryPartitionSizeInBytes**..	256MB
spark.sql.adaptive.shuffle.targetPostShuffleInputSize	Minimum amount of shuffle data processed by each task. The unit is byte.	67108864

24.12.3 Optimizing Spark SQL Performance in the Small File Scenario

Scenario

A Spark SQL table may have many small files (far smaller than an HDFS block), each of which maps to a partition on the Spark by default. In other words, each small file is a task. If the small files are great in number, Spark must initiate a large number of tasks. If shuffle operations exist in Spark SQL, the number of hash buckets increases, affecting performance.

In this scenario, you can manually specify the split size of each task to avoid an excessive number of tasks and improve performance.

NOTE

If the SQL logic does not involve shuffle operations, this optimization does not improve performance.

Configuration

If you want to enable small file optimization, configure the **spark-defaults.conf** file on the Spark client.

Table 24-28 Parameter description

Parameter	Description	Default Value
spark.sql.files.maxPartitionBytes	The maximum number of bytes that can be packed into a single partition when a file is read. Unit: byte	134217728 (128 MB)
spark.files.openCostInBytes	The estimated cost to open a file, measured by the number of bytes that can be scanned in the same time. This is used when putting multiple files into a partition. It is better to over estimate, then the partitions with small files will be faster than partitions with larger files.	4 MB

24.12.4 Optimizing the Spark INSERT SELECT Statement

Scenario

The **INSERT...SELECT** operation can be optimized in the following scenarios:

- Data in a large number of small files is queried.
- Data in large files is queried.
- A non-Spark user is used in Beeline/JDBCServer mode.

Procedure

The **INSERT...SELECT** operation can be optimized as follows:

- When creating a Hive table, set the storage type to Parquet to accelerate execution of the **INSERT...SELECT** statement.
- Use **spark-sql** or a Spark user in Beeline/JDBCServer mode to execute **INSERT...SELECT** operations. This eliminates the need for changing the file owner, which quickens **INSERT...SELECT** statement execution.

  NOTE

  In Beeline/JDBCServer mode, an executor and a driver are run by the same user. Because a driver is a part of JDBCServer and JDBCServer is run by a Spark user, the driver is also run by the Spark user. At present, the user of the Beeline client cannot be transparently transmitted to the executor during operation. If a non-Spark user is used, the owner of a file must be changed to the user of the Beeline client, that is, the actual user.
- Querying a large number of small files triggers multiple map operations, which generates numerous small output files that are time-consuming to rename. To tackle this problem, you can set **spark.sql.files.maxPartitionBytes** and **spark.files.openCostInBytes** to limit the number of bytes read by a partition. This will consolidate multiple small files in a partition, reducing the number of output files and the time needed to rename them. Ultimately, this will shorten the execution time of the **INSERT...SELECT** statement.

NOTE

These optimizations are ineffective in the following scenario:

A dynamic partition table has a large number of partitions.

24.12.5 Configuring Multiple Concurrent Clients to Connect to JDBCServer

Scenario

Multiple clients can be connected to JDBCServer at the same time. However, if the number of concurrent tasks is too large, the default configuration of JDBCServer must be optimized to adapt to the scenario.

Procedure

1. Set the fair scheduling policy of JDBCServer.

   The default scheduling policy of Spark is **FIFO**, which may cause a failure of short tasks in multi-task scenarios. Therefore, the fair scheduling policy must be used in multi-task scenarios to prevent task failure.

   a. For details about how to configure Fair Scheduler in Spark, visit **https://archive.apache.org/dist/spark/docs/3.1.1/job-scheduling.html#scheduling-within-an-application**.

   b. Configure Fair Scheduler on the JDBC client.

i. Run the following statement in either the Beeline CLI or custom JDBC code. The **PoolName** parameter indicates a scheduling pool for fair scheduling.

```
SET spark.sql.thriftserver.scheduler.pool=PoolName;
```

ii. Run the SQL command. The Spark task will be executed in the preceding scheduling pool.

2. Set the **BroadCastHashJoin** timeout interval.

   There is a timeout parameter of **BroadCastHashJoin**. The task query fails if the query period exceeds the preset timeout interval. In multi-task scenarios, the Spark task of BroadCastHashJoin may fail due to resource preemption. Therefore, it is necessary to modify the timeout interval in the **spark-defaults.conf** file of JDBCServer.

   Table 24-29 Parameter description

Parameter	Description	Default Value
spark.sql.broadcastTimeout	The timeout interval in the broadcast table of **BroadcastHashJoin**. If there are many concurrent tasks, set the parameter to a larger value or a negative number.	-1 (Numeral type. The actual value is 5 minutes.)

24.12.6 Configuring the Default Number of Data Blocks Divided by SparkSQL

Scenarios

By default, SparkSQL divides data into 200 data blocks during shuffle. In data-intensive scenarios, each data block may have excessive size. If a single data block of a task is larger than 2 GB, an error similar to the following will be reported while Spark attempts to fetch the data block:

```
Adjusted frame length exceeds 2147483647: 2717729270 - discarded
```

For example, setting the number of default data blocks to 200 causes SparkSQL to encounter an error in running a TPCDS 500-GB test. To avoid this, increase the number of default blocks in data-intensive scenarios.

Configuration parameters

Navigation path for setting parameters:

On Manager, choose **Cluster** > **Services** > **Spark2x**, click **Configurations**, and click **All Configurations**. Enter a parameter name in the search box.

Table 24-30 Parameter description

Parameter	Description	Default Value
spark.sql.shuffle.partitions	Indicates the default number of blocks divided during shuffle.	200

24.12.7 Optimizing Memory When Data Is Inserted into Spark Dynamic Partitioned Tables

Scenario

When Spark SQL inserts data into dynamic partitioned tables, the more partitions there are, the more HDFS files a single task generates and the more memory metadata occupies. In this case, GC is severe and OOM may occur.

Based on the test results, there are 10,240 tasks and 2,000 partitions. Before renaming an HDFS file from the temporary to the target directory, the FileStatus metadata is approximately 29 GB in size. To avoid the preceding problem, you can modify the SQL statement to repartition data to reduce the number of HDFS files.

Procedure

Insert **distribute by** followed by partition fields into dynamic partition statements.

Example:

```
insert into table store_returns partition (sr_returned_date_sk) select
sr_return_time_sk,sr_item_sk,sr_customer_sk,sr_cdemo_sk,sr_hdemo_sk,sr_addr_sk,sr_store_sk,sr_reason_sk,sr
_ticket_number,sr_return_quantity,sr_return_amt,sr_return_tax,sr_return_amt_inc_tax,sr_fee,sr_return_ship_co
st,sr_refunded_cash,sr_reversed_charge,sr_store_credit,sr_net_loss,sr_returned_date_sk from $
{SOURCE}.store_returns distribute by sr_returned_date_sk;
```

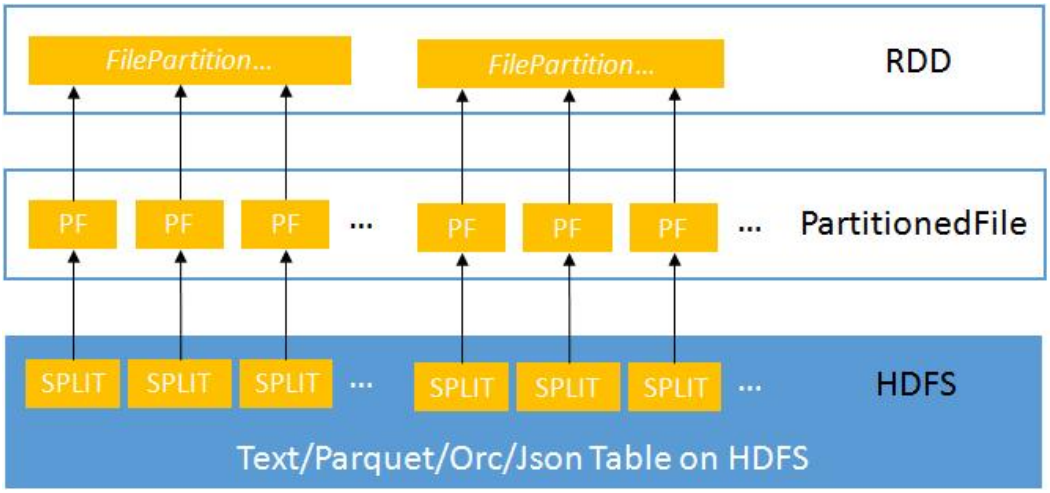
24.12.8 Optimizing Small Files

Scenario

A Spark SQL table may have many small files (far smaller than an HDFS block), each of which maps to a partition on the Spark by default. In other words, each small file is a task. In this way, Spark has to start many such tasks. If a shuffle operation is involved in the SQL logic, the number of hash buckets soars, severely hindering system performance.

In case of massive number of small files, when DataSource creates an RDD, it splits small files in the Spark SQL table to PartitionedFiles and then merges the PartitionedFiles to a partition to avoid generating too many hash buckets during the shuffle operation. See **Figure 24-8**.

Figure 24-8 Merging small files



Procedure

If you want to enable small file optimization, configure the **spark-defaults.conf** file on the Spark client.

Table 24-31 Parameter description

Parameter	Description	Default Value
spark.sql.files.maxPartitionBytes	The maximum number of bytes that can be packed into a single partition when a file is read. Unit: byte	134217728 (128 MB)
spark.files.openCostInBytes	The estimated cost to open a file, measured by the number of bytes that can be scanned in the same time. This is used when putting multiple files into a partition. It is better to over estimate, then the partitions with small files will be faster than partitions with larger files.	4 MB

24.12.9 Optimizing the Aggregate Algorithms

Scenario

Spark SQL supports hash aggregate algorithm. Namely, use fast aggregate hashmap as cache to improve aggregate performance. The hashmap replaces the previous ColumnarBatch to avoid performance problems caused by the wide mode (multiple key or value fields) of an aggregate table.

Procedure

If you want to enable optimization of aggregate algorithm, configure following parameters in the **spark-defaults.conf** file on the Spark client.

Table 24-32 Parameter description

Parameter	Description	Default Value
spark.sql.codegen.aggregate.map.twolevel.enabled	Specifies whether to enable aggregation algorithm optimization. • **true**: Enable • **false**: Disable	true

24.12.10 Optimizing Datasource Tables

Scenario

Save the partition information about the datasource table to the Metastore and process partition information in the Metastore.

- Optimize the datasource tables, support syntax such as adding, deletion, and modification in the table based on partitions, improving compatibility with Hive.
- Support statements of partition tailoring and push down to the Metastore to filter unmatched partitions.

  Example:

  ```
  select count(*) from table where partCol=1;    //partCol (partition column)
  ```

  You need only to process data corresponding to partCol=1 when performing the TableScan operation in the physical plan.

Procedure

If you want to enable Datasource table optimization, configure the **spark-defaults.conf** file on the Spark client.

Table 24-33 Parameter description

Parameter	Description	Default Value
spark.sql.hive.manageFilesourcePartitions	Specifies whether to enable Metastore partition management (including datasource tables and converted Hive). • **true** indicates enabling Metastore partition management. In this case, datasource tables are stored in Hive and Metastore is used to tailor partitions in query statements. • **false** indicates disabling Metastore partition management.	true
spark.sql.hive.metastorePartitionPruning	Specifies whether to support pushing down predicate to Hive Metastore. • **true** indicates supporting pushing down predicate to Hive Metastore. Only the predicate of Hive tables is supported. • **false** indicates not supporting pushing down predicate to Hive Metastore.	true
spark.sql.hive.filesourcePartitionFileCacheSize	The cache size of the partition file metadata in the memory. All tables share a cache that can use up to specified num bytes for file metadata. This parameter is valid only when **spark.sql.hive.manageFilesourcePartitions** is set to **true**.	250 * 1024 * 1024
spark.sql.hive.convertMetastoreOrc	The processing approach of ORC tables. • **false**: Spark SQL uses Hive SerDe to process ORC tables. • **true**: Spark SQL uses the Spark built-in mechanism to process ORC tables.	true

24.12.11 Merging CBO

Scenario

Spark SQL supports rule-based optimization by default. However, the rule-based optimization cannot ensure that Spark selects the optimal query plan. Cost-Based Optimizer (CBO) is a technology that intelligently selects query plans for SQL statements. After CBO is enabled, the CBO optimizer performs a series of

estimations based on the table and column statistics to select the optimal query plan.

Procedure

Perform the following steps to enable CBO:

1. You need to run corresponding SQL commands to collect required table and column statistics.

   SQL commands are as follows (to be chosen as required):

   - Generate table-level statistics (table scanning):

     ANALYZE TABLE src COMPUTE STATISTICS

     This command generates **sizeInBytes** and **rowCount**.

     When you use the ANALYZE statement to collect statistics, sizes of tables not from HDFS cannot be calculated.

   - Generate table-level statistics (no table scanning):

     ANALYZE TABLE src COMPUTE STATISTICS NOSCAN

     This command generates only **sizeInBytes**. Compared with the originally generated **sizeInBytes** and **rowCount** if the **sizeInBytes** remains unchanged, **rowCount** (if any) reserves. Otherwise, **rowCount** is cleared.

   - Generate column-level statistics:

     ANALYZE TABLE src COMPUTE STATISTICS FOR COLUMNS a, b, c

     This command generates column statistics and updates table statistics for consistency. Statistics of complicated data types (such as Seq and Map) and HiveStringType cannot be generated.

   - Display statistics:

     DESC FORMATTED src

     This command displays *xxx* bytes and *xxx* rows in **Statistics** to indicate table-level statistics. You can also run the following command to display column statistics:

     DESC FORMATTED src a

   Limitation: The current statistics collection does not support statistics for partition levels for partitioned tables.

2. Configure parameters in **Table 24-34** in the **spark-defaults.conf** file on the Spark client.

   Table 24-34 Parameter description

Parameter	Description	Default Value
spark.sql.cbo.enabled	The switch to enable or disable CBO. • **true**: Enable • **false**: Disable To enable this function, ensure that statistics of related tables and columns are generated.	false

Parameter	Description	Default Value
spark.sql.cbo.joinReorder.enabled	Specifies whether to automatically adjust the sequence of consecutive inner joins by using CBO. • **true**: Enable • **false**: Disable To enable this function, ensure that statistics of related tables and columns are generated and CBO is enabled.	false
spark.sql.cbo.joinReorder.dp.threshold	Specifies the threshold of the number of tables that the sequence of consecutive inner joins is automatically adjusted by CBO. If the threshold is exceeded, the sequence of joins is not adjusted.	12

24.12.12 SQL Optimization for Multi-level Nesting and Hybrid Join

Scenario

This section describes the optimization suggestions for SQL statements in multi-level nesting and hybrid join scenarios.

Prerequisites

The following provides an example of complex query statements:

```
select
s_name,
count(1) as numwait
from (
select s_name from (
select
s_name,
t2.l_orderkey,
l_suppkey,
count_suppkey,
max_suppkey
from
test2 t2 right outer join (
select
s_name,
l_orderkey,
l_suppkey from (
select
s_name,
t1.l_orderkey,
l_suppkey,
count_suppkey,
max_suppkey
from
```



```
test1 t1 join (
select
s_name,
l_orderkey,
l_suppkey
from
orders o join (
select
s_name,
l_orderkey,
l_suppkey
from
nation n join supplier s
on
s.s_nationkey = n.n_nationkey
and n.n_name = 'SAUDI ARABIA'
join lineitem l
on
s.s_suppkey = l.l_suppkey
where
l.l_receiptdate > l.l_commitdate
and l.l_orderkey is not null
) l1 on o.o_orderkey = l1.l_orderkey and o.o_orderstatus = 'F'
) l2 on l2.l_orderkey = t1.l_orderkey
) a
where
(count_suppkey > 1)
or ((count_suppkey=1)
and (l_suppkey <> max_suppkey))
) l3 on l3.l_orderkey = t2.l_orderkey
) b
where
(count_suppkey is null)
or ((count_suppkey=1)
and (l_suppkey = max_suppkey))
) c
group by
s_name
order by
numwait desc,
s_name
limit 100;
```

Procedure

Step 1 Analyze services.

Analyze business to determine whether SQL statements can be simplified through measures, for example, by combining tables to reduce the number of nesting levels layers and join times.

Step 2 If the SQL statements cannot be simplified, configure the driver memory.

- Use **spark-submit** or **spark-sql** to run SQL statements and go to **Step 3**.
- Use **spark-beeline** to run SQL statements and go to **Step 4**.

Step 3 During execution of SQL statements, specify the **driver-memory** parameter. An example of SQL statements is as follows:

/spark-sql --master=local[4] --driver-memory=512M -f /tpch.sql

Step 4 Before you run SQL statements, change the memory size as the MRS cluster administrator.

1. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark2x**, and click **Configurations**.

2. Click the **All Configurations** sub-tab and search for **SPARK_DRIVER_MEMORY**.
3. Set the parameter to a larger value to increase the memory size. The value must be an integer, and the unit must be MB or GB. For example, enter **512 MB**.

----End

Related Information

In the event of insufficient DRIVER memory, the following error may be displayed during the query:

```
2018-02-11 09:13:14,683 | WARN  | Executor task launch worker for task 5 | Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0. |
org.apache.spark.sql.catalyst.expressions.RowBasedKeyValueBatch.spill(RowBasedKeyValueBatch.java:173)
2018-02-11 09:13:14,682 | WARN  | Executor task launch worker for task 3 | Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0. |
org.apache.spark.sql.catalyst.expressions.RowBasedKeyValueBatch.spill(RowBasedKeyValueBatch.java:173)
2018-02-11 09:13:14,704 | ERROR | Executor task launch worker for task 2 | Exception in task 2.0 in stage
1.0 (TID 2) | org.apache.spark.internal.Logging$class.logError(Logging.scala:91)
java.lang.OutOfMemoryError: Unable to acquire 262144 bytes of memory, got 0
        at org.apache.spark.memory.MemoryConsumer.allocateArray(MemoryConsumer.java:100)
        at org.apache.spark.unsafe.map.BytesToBytesMap.allocate(BytesToBytesMap.java:791)
        at org.apache.spark.unsafe.map.BytesToBytesMap.<init>(BytesToBytesMap.java:208)
        at org.apache.spark.unsafe.map.BytesToBytesMap.<init>(BytesToBytesMap.java:223)
        at
org.apache.spark.sql.execution.UnsafeFixedWidthAggregationMap.<init>(UnsafeFixedWidthAggregationMap.j
ava:104)
        at
org.apache.spark.sql.execution.aggregate.HashAggregateExec.createHashMap(HashAggregateExec.scala:307)
        at org.apache.spark.sql.catalyst.expressions.GeneratedClass
$GeneratedIterator.agg_doAggregateWithKeys$(Unknown Source)
        at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIterator.processNext(Unknown
Source)
        at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)
        at org.apache.spark.sql.execution.WholeStageCodegenExec$$anonfun$8$$anon
$1.hasNext(WholeStageCodegenExec.scala:381)
        at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:408)
        at
org.apache.spark.shuffle.sort.BypassMergeSortShuffleWriter.write(BypassMergeSortShuffleWriter.java:126)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:96)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:53)
        at org.apache.spark.scheduler.Task.run(Task.scala:99)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:325)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
```

24.13 Spark Streaming Performance Tuning

Scenario

Streaming is a mini-batch streaming processing framework that features second-level delay and high throughput. To optimize Streaming is to improve its throughput while maintaining second-level delay so that more data can be processed per unit time.

NOTE

This section applies to the scenario where the input data source is Kafka.

Procedure

A simple streaming processing system consists of a data source, a receiver, and a processor. The data source is Kafka, the receiver is the Kafka data source receiver of Streaming, and the processor is Streaming.

Enhance the performance of the three components for Streaming tuning.

- **Data source optimization**

  In actual application scenarios, the data source stores the data in the local disks to ensure the error tolerance of the data. However, the calculation results of the Streaming are stored in the memory, and the data source may become the largest bottleneck of the streaming system.

  Kafka can be optimized from the following aspects:

  - Use Kafka-0.8.2 or later version that allows you to use new Producer APIs in asynchronous mode.
  - Configure multiple Broker directories, multiple I/O threads, and a proper number of partitions for a topic.

  For details, see section **Performance Tuning** in the Kafka open source documentation at **http://kafka.apache.org/documentation.html**.

- **Receiver optimization**

  Streaming has multiple data source receivers, such as Kafka, Flume, MQTT, and ZeroMQ. Kafka has the most receiver types and is the most mature receiver.

  Kafka provides three types of receiver APIs:

  - KafkaReceiver directly receives Kafka data. If the process is abnormal, data may be lost.
  - ReliableKafkaReceiver receives data displacement through ZooKeeper records.
  - DirectKafka reads data from each partition of Kafka through the RDD, ensuring high reliability.

  According to the implementation mechanism and test results, DirectKafka provides better performance than the other two APIs. Therefore, the DirectKafka API is recommended to implement the receiver.

  Kafka receivers function as Kafka consumers. For details about how to optimize them, see the Kafka open source documentation at **http://kafka.apache.org/documentation.html**.

- **Processor optimization**

  The bottom layer of Spark Streaming is executed by Spark. Therefore, most optimization measures for Spark can also be applied to Spark Streaming. The following is an example:

  - Data serialization
  - Memory configuration
  - Configuring DOP
  - Using the external shuffle service to improve performance

NOTE

Higher performance of Spark Streaming indicates lower overall reliability. Examples:

If **spark.streaming.receiver.writeAheadLog.enable** is set to **false**, disk I/Os are reduced and performance is improved. However, because WAL is disabled, data is lost during fault recovery.

Therefore, do not disable configuration items that ensure data reliability in production environments during Spark Streaming tuning.

- **Log archive optimization**

  The **spark.eventLog.group.size** parameter is used to group **JobHistory** logs of an application based on the specified number of jobs. Each group creates a file recording log to prevent **JobHistory** reading failures caused by an oversized log generated during the long-term running of the application. If this parameter is set to **0**, logs are not grouped.

  Most Spark Streaming jobs are small jobs and are generated at a high speed. As a result, frequent grouping is performed and a large number of small log files are generated, consuming disk I/O resources. You are advised to increase the parameter value to, for example, **1000** or greater.

24.14 Spark on OBS Performance Tuning

Scenario

In the scenario where a small number of requests are frequently sent from Spark on OBS to OBS, you can disable OBS monitoring to improve performance.

Configuration

Modify the configuration in the **core-site.xml** file on the Spark client.

Table 24-35 Parameter description

Parameter	Description	Default Value
fs.obs.metrics.switch	Specifies whether to report OBS monitoring metrics. • **true**: enable • **false**: disable	true

Parameter	Description	Default Value
fs.obs.metrics.consumer	Specifies the processing mode of OBS monitoring metrics. • **org.apache.hadoop.fs.obs.metrics.OBSAMetricsProvider**: indicates that OBS monitoring metrics are collected. • **org.apache.hadoop.fs.obs.DefaultMetricsConsumer**: indicates that OBS monitoring metrics are not collected. To use the OBS monitoring function, ensure that the function of reporting OBS monitoring metrics is enabled.	org.apache.hadoop.fs.obs.metrics.OBSAMetricsProvider

24.15 Spark O&M Management

24.15.1 Configuring Spark Parameters Rapidly

Overview

This section describes how to quickly configure common parameters and lists parameters that are not recommended to be modified when Spark2x is used.

Common parameters to be configured

Some parameters have been adapted during cluster installation. However, the following parameters need to be adjusted based on application scenarios. Unless otherwise specified, the following parameters are configured in the **spark-defaults.conf** file on the Spark2x client.

Table 24-36 Common parameters to be configured

Configuration Item	Description	Default Value
spark.sql.parquet.compression.codec	Used to set the compression format of a non-partitioned Parquet table. Set the queue in the **spark-defaults.conf** configuration file on the JDBCServer server.	snappy

Configuration Item	Description	Default Value
spark.dynamicAllocation.enabled	Indicates whether to use dynamic resource scheduling, which is used to adjust the number of executors registered with the application according to scale. Currently, this parameter is valid only in Yarn mode. The default value for JDBCServer is **true**, and that for the client is **false**.	false
spark.executor.memory	Indicates the memory size used by each executor process. Its character sting is in the same format as the JVM memory (example: 512 MB or 2 GB).	4G
spark.sql.autoBroadcastJoinThreshold	Indicates the maximum value for the broadcast configuration when two tables are joined. ● When the size of a field in a table involved in an SQL statement is less than the value of this parameter, the system broadcasts the SQL statement. ● If the value is set to **-1**, broadcast is not performed.	10485760
spark.yarn.queue	Specifies the Yarn queue where JDBCServer resides. Set the queue in the **spark-defaults.conf** configuration file on the JDBCServer server.	default
spark.driver.memory	In a large cluster, you are advised to configure the memory used by the 32 GB to 64 GB driver process, that is, the SparkContext initialization process (for example, 512 MB and 2 GB).	4G
spark.yarn.security.credentials.hbase.enabled	Indicates whether to enable the function of obtaining HBase tokens. If the Spark on HBase function is required and a security cluster is configured, set this parameter to **true**. Otherwise, set this parameter to **false**.	false

Configuration Item	Description	Default Value
spark.serializer	Used to serialize the objects that are sent over the network or need to be cached. The default value of Java serialization applies to any Serializable Java object, but the running speed is slow. Therefore, you are advised to use **org.apache.spark.serializer.KryoSerializer** and configure Kryo serialization. It can be any subclass of **org.apache.spark.serializer.Serializer**.	org.apache.spark.serializer.JavaSerializer
spark.executor.cores	Indicates the number of kernels used by each executor. Set this parameter in standalone mode and Mesos coarse-grained mode. When there are sufficient kernels, the application is allowed to execute multiple executable programs on the same worker. Otherwise, each application can run only one executable program on each worker.	1
spark.shuffle.service.enabled	Indicates a long-term auxiliary service in NodeManager for improving shuffle computing performance.	false
spark.sql.adaptive.enabled	Indicates whether to enable the adaptive execution framework.	false
spark.executor.memoryOverhead	Indicates the heap memory to be allocated to each executor, in MB. This is the memory that occupies the overhead of the VM, similar to the internal string and other built-in overhead. The value increases with the executor size (usually 6% to 10%).	1 GB
spark.streaming.kafka.direct.lifo	Indicates whether to enable the LIFO function of Kafka.	false

Parameters Not Recommended to Be Modified

The following parameters have been adapted during cluster installation. You are not advised to modify them.

Table 24-37 Parameters not recommended to be modified

Configuration Item	Description	Default Value or Configuration Example
spark.password.factory	Selects the password parsing mode.	org.apache.spark.om.util.FIPasswordFactory
spark.ssl.ui.protocol	Sets the SSL protocol of the UI.	TLSv1.2
spark.yarn.archive	Archives Spark JAR files, which are distributed to Yarn cache. If this parameter is set, the value will replace **<code> spark.yarn.jars </code>** and be archived in the containers of all applications. The archive should contain the JAR files in its root directory. Archives can also be hosted on HDFS to speed up file distribution.	hdfs://hacluster/user/spark2x/jars/xxx/spark-archive-2x.zip **NOTE** The version xxx is used as an example. Replace it with the actual version number.
spark.yarn.am.extraJavaOptions	Indicates a string of extra JVM options to pass to the YARN ApplicationMaster in client mode. Use **spark.driver.extraJavaOptions** in cluster mode.	-Dlog4j.configuration=./__spark_conf__/__hadoop_conf__/log4j-executor.properties -Djava.security.auth.login.config=./__spark_conf__/__hadoop_conf__/jaas-zk.conf -Dzookeeper.server.principal=zookeeper/hadoop.<*system domain name*> -Djava.security.krb5.conf=./__spark_conf__/__hadoop_conf__/kdc.conf -Djdk.tls.ephemeralDHKeySize=2048
spark.shuffle.servicev2.port	Indicates the port for the shuffle service to monitor requests for obtaining data.	27338
spark.ssl.historyServer.enabled	Sets whether the history server uses SSL.	true

Configuration Item	Description	Default Value or Configuration Example
spark.files.overwrite	When the target file exists and its content does not match that of the source file, whether to overwrite the file added through **SparkContext.addFile()**.	false
spark.yarn.cluster.driver.extraClassPath	Indicates the extraClassPath of the driver in Yarn-cluster mode. Set the parameter to the path and parameters of the server.	${BIGDATA_HOME}/common/runtime/security
spark.driver.extraClassPath	Indicates the extra class path entries attached to the class path of the driver.	${BIGDATA_HOME}/common/runtime/security
spark.yarn.dist.innerfiles	Sets the files that need to be uploaded to HDFS from Spark in Yarn mode.	/Spark_path/spark/conf/s3p.file,/Spark_path/spark/conf/locals3.jceks *Spark_path* is the installation path of the Spark client.
spark.sql.bigdata.register.dialect	Registers the SQL parser.	org.apache.spark.sql.hbase.HBaseSQLParser
spark.shuffle.manager	Indicates the data processing mode. There are two implementation modes: sort and hash. The sort shuffle has a higher memory utilization. It is the default option in Spark 1.2 and later versions. Spark 2.*x* and later versions do not support hash.	SORT

Configuration Item	Description	Default Value or Configuration Example
spark.deploy.zookeeper.url	Indicates the address of ZooKeeper. Multiple addresses are separated by commas (,).	For example: host1:2181,host2:2181,host3:2181
spark.broadcast.factory	Indicates the broadcast mode.	org.apache.spark.broadcast.TorrentBroadcastFactory
spark.sql.session.state.builder	Session state constructor.	org.apache.spark.sql.hive.FIHiveACLSessionStateBuilder
spark.executor.extraLibraryPath	Sets the special library path used when the executor JVM is started.	${BIGDATA_HOME}/FusionInsight_HD_xxx/install/FusionInsight-Hadoop-*/hadoop/lib/native
spark.ui.customErrorPage	Indicates whether to display the custom error information page when an error occurs on the page.	true
spark.httpdProxy.enable	Indicates whether to use the httpd proxy.	true
spark.ssl.ui.enabledAlgorithms	Sets the SSL algorithm of UI.	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_DHE_DSS_WITH_AES_256_GCM_SHA384,TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
spark.ui.logout.enabled	Sets the logout button for the web UI of the Spark component.	true
spark.security.hideInfo.enabled	Indicates whether to hide sensitive information on the UI.	true

Configuration Item	Description	Default Value or Configuration Example
spark.yarn.cluster.driver.extraLibraryPath	Indicates the **extraLibraryPath** of the driver in Yarn-cluster mode. Set this parameter to the path and parameters of the server.	${BIGDATA_HOME}/FusionInsight_HD_xxx/install/FusionInsight-Hadoop-*/hadoop/lib/native
spark.driver.extraLibraryPath	Sets a special library path for starting the driver JVM.	${DATA_NODE_INSTALL_HOME}/hadoop/lib/native
spark.ui.killEnabled	Allows stages and jobs to be stopped on the web UI.	true
spark.yarn.access.hadoopFileSystems	Spark can access multiple NameService instances. If there are multiple NameService instances, set this parameter to all the NameService instances and separate them with commas (,).	hdfs://hacluster,hdfs://hacluster

Configuration Item	Description	Default Value or Configuration Example
spark.yarn.cluster.driver.extraJavaOptions	Indicates extra JVM option passed to the executor, for example, GC setting and logging. Do not set Spark attributes or heap size using this option. Instead, set Spark attributes using the SparkConf object or the **spark-defaults.conf** file specified when the spark-submit script is called. Set heap size using **spark.executor.memory**.	-Xloggc:<LOG_DIR>/gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -Dlog4j.configuration=./__spark_conf__/__hadoop_conf__/log4j-executor.properties -Djava.security.auth.login.config=./__spark_conf__/__hadoop_conf__/jaas-zk.conf -Dzookeeper.server.principal=zookeeper/hadoop.*<system domain name>* -Djava.security.krb5.conf=./__spark_conf__/__hadoop_conf__/kdc.conf -Djetty.version=x.y.z -Dorg.xerial.snappy.tempdir=${BIGDATA_HOME}/tmp/spark2x_app -Dcarbon.properties.filepath=./__spark_conf__/__hadoop_conf__/carbon.properties -Djdk.tls.ephemeralDHKeySize=2048
spark.driver.extraJavaOptions	Indicates a series of extra JVM options passed to the driver,	-Xloggc:${SPARK_LOG_DIR}/indexserver-omm-%p-gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:MaxDirectMemorySize=512M -XX:MaxMetaspaceSize=512M -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -XX:OnOutOfMemoryError='kill -9 %p' -Djetty.version=x.y.z -Dorg.xerial.snappy.tempdir=${BIGDATA_HOME}/tmp/spark2x/JDBCServer/snappy_tmp -Djava.io.tmpdir=${BIGDATA_HOME}/tmp/spark2x/JDBCServer/io_tmp -Dcarbon.properties.filepath=${SPARK_CONF_DIR}/carbon.properties -Djdk.tls.ephemeralDHKeySize=2048 -Dspark.ssl.keyStore=${SPARK_CONF_DIR}/child.keystore #{java_stack_prefer}
spark.eventLog.overwrite	Indicates whether to overwrite any existing file.	false

Configuration Item	Description	Default Value or Configuration Example
spark.eventLog.dir	Indicates the directory for logging Spark events if **spark.eventLog.enabled** is set to **true**. In this directory, Spark creates a subdirectory for each application and logs events of the application in the subdirectory. You can also set a unified address similar to the HDFS directory so that the History Server can read historical files.	hdfs://hacluster/spark2xJobHistory2x
spark.random.port.min	Sets the minimum random port.	22600
spark.authenticate	Indicates whether Spark authenticates its internal connections. If the application is not running on Yarn, see **spark.authenticate.secret**.	true
spark.random.port.max	Sets the maximum random port.	22899
spark.eventLog.enabled	Indicates whether to log Spark events, which are used to reconstruct the web UI after the application execution is complete.	true

Configuration Item	Description	Default Value or Configuration Example
spark.executor.extraJavaOptions	Indicates extra JVM option passed to the executor, for example, GC setting and logging. Do not set Spark attributes or heap size using this option.	-Xloggc:<LOG_DIR>/gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -Dlog4j.configuration=./log4j-executor.properties -Djava.security.auth.login.config=./jaas-zk.conf -Dzookeeper.server.principal=zookeeper/hadoop.*<system domain name>* -Djava.security.krb5.conf=./kdc.conf -Dcarbon.properties.filepath=./carbon.properties -Xloggc:<LOG_DIR>/gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -Dlog4j.configuration=./__spark_conf__/__hadoop_conf__/log4j-executor.properties -Djava.security.auth.login.config=./__spark_conf__/__hadoop_conf__/jaas-zk.conf -Dzookeeper.server.principal=zookeeper/hadoop.*<system domain name>* -Djava.security.krb5.conf=./__spark_conf__/__hadoop_conf__/kdc.conf -Dcarbon.properties.filepath=./__spark_conf__/__hadoop_conf__/carbon.properties -Djdk.tls.ephemeralDHKeySize=2048
spark.sql.authorization.enabled	Indicates whether to enable authentication for the Hive client.	true

24.15.2 Spark Common Configuration Parameters

Overview

This section describes common configuration items used in Spark. Subsections are divided by feature so that you can quickly find required configuration items. If you

use MRS clusters, most parameters described in this section have been adapted and you do not need to configure them again. For details about the parameters that need to be configured based on the site requirements, see **Configuring Spark Parameters Rapidly**.

Configuring the Number of Stage Retries

When FetchFailedException occurs in a Spark task, a stage retry is triggered. To prevent infinite stage retries, the number of stage retries is limited. The number of retry times can be adjusted based on the site requirements.

Configure the following parameters in the **spark-defaults.conf** file on the Spark client.

Table 24-38 Parameter description

Parameter	Description	Default Value
spark.stage.maxConsecutiveAttempts	Indicates the maximum number of stage retries.	4

Configuring Whether to Use Cartesian Product

To enable the Cartesian product function, configure the following parameter in the **spark-defaults.conf** configuration file of Spark.

Table 24-39 Cartesian product parameters

Parameter	Description	Default Value
spark.sql.crossJoin.enabled	Indicates whether to allow implicit Cartesian product execution. • **true**: Implicit Cartesian product execution is allowed. • **false**: Implicit Cartesian product execution is not allowed. In this case, only CROSS JOIN can be explicitly included in the query.	true

NOTE

- For JDBC applications, configure this parameter in the **spark-defaults.conf** configuration file of the server.
- For tasks submitted by the Spark client, configure this parameter in the **spark-defaults.conf** configuration file of the client.

Configuring Security Authentication for Long-Time Spark Tasks

In security mode, if the **kinit** command is used for security authentication when the Spark CLI (such as spark-shell, spark-sql, or spark-submit) is used, the task fails due to authentication expiration when the task is running for a long time.

Set the following parameters in the **spark-defaults.conf** configuration file on the client. After the configuration is complete, run the Spark CLI again.

NOTE

If this parameter is set to **true**, ensure that the values of **keytab** and **principal** in **spark-defaults.conf** and **hive-site.xml** are the same.

Table 24-40 Parameter description

Parameter	Description	Default Value
spark.kerberos.principal	Indicates the principal user who has the Spark operation permission. Contact the MRS cluster administrator to obtain the principal user.	-
spark.kerberos.keytab	Indicates the name and path of the keytab file used to configure Spark operation permissions. Contact the MRS cluster administrator to obtain the Keytab file.	-
spark.security.bigdata.loginOnce	Indicates whether the principal user logs in to the system only once. **true**: single login; **false**: multiple logins. The difference between a single login and multiple logins is as follows: The Spark community uses the Kerberos user to log in to the system for multiple times. However, the TGT or token may expire, causing the application to fail to run for a long time. The Kerberos login mode of DataSight is modified to allow users to log in only once, which effectively resolves the expiration problem. The restrictions are as follows: The principal and keytab configuration items of Hive must be the same as those of Spark. **NOTE** If this parameter is set to **true**, ensure that the values of **keytab** and **principal** in **spark-defaults.conf** and **hive-site.xml** are the same.	true

Python Spark

Python Spark is the third programming language of Spark except Scala and Java. Different from Java and Scala that run on the JVM platform, Python Spark has its own Python process as well as the JVM process. The following configuration items

apply only to Python Spark scenarios. However, other configuration items can also take effect in Python Spark scenarios.

Table 24-41 Parameter description

Parameter	Description	Default Value
spark.python.profile	Indicates whether to enable profiling on the Python worker. Use **sc.show_profiles()** to display the analysis results or display the analysis results before the Driver exits. You can use **sc.dump_profiles(path)** to dump the results to a disk. If some analysis results have been manually displayed, they will not be automatically displayed before the driver exits. By default, **pyspark.profiler.BasicProfiler** is used. You can transfer the specified profiler during SparkContext initialization to overwrite the default profiler.	false
spark.python.worker.memory	Indicates the memory size that can be used by each Python worker process during aggregation. The value format is the same as that of the specified JVM memory, for example, 512 MB and 2 GB. If the memory used by a process during aggregation exceeds the value of this parameter, data will be written to disks.	512m
spark.python.worker.reuse	Indicates whether to reuse Python workers. If the reuse function is enabled, a fixed number of Python workers will be reused by the next batch of submitted tasks instead of forking a Python process for each task. This function is useful in large-scale broadcasting because the data does not need to be transferred from the JVM to the Python workers again for the next batch of submitted tasks.	true

Dynamic Allocation

Dynamic resource scheduling is a unique feature of the On Yarn mode. This function can be used only after Yarn External Shuffle is enabled. When Spark is used as a resident service, dynamic resource scheduling greatly improves resource utilization. For example, the JDBCServer process does not accept JDBC requests in most of the time. Therefore, releasing resources in this period greatly reduces the waste of cluster resources.

Table 24-42 Parameter description

Parameter	Description	Default Value
spark.dynamicAllocation.enabled	Indicates whether to use dynamic resource scheduling, which is used to adjust the number of executors registered with the application according to scale. Currently, this parameter is valid only in YARN mode. To enable dynamic resource scheduling, set **spark.shuffle.service.enabled** to **true**. Related parameters are as follows: **spark.dynamicAllocation.minExecutors**, **spark.dynamicAllocation.maxExecutors**, and **spark.dynamicAllocation.initialExecutors**.	• JDBCServer2x: true • SparkResource2x: false
spark.dynamicAllocation.minExecutors	Indicates the minimum number of executors.	0
spark.dynamicAllocation.initialExecutors	Indicates the number of initial executors.	spark.dynamicAllocation.minExecutors
spark.dynamicAllocation.maxExecutors	Indicates the maximum number of executors.	2048
spark.dynamicAllocation.schedulerBacklogTimeout	Indicates the first timeout period for scheduling. The unit is second.	1s
spark.dynamicAllocation.sustainedSchedulerBacklogTimeout	Indicates the second and later timeout interval for scheduling.	1s
spark.dynamicAllocation.executorIdleTimeout	Indicates the idle timeout interval for common executors. The unit is second.	60

Parameter	Description	Default Value
spark.dynamicAllocation.cachedExecutorIdleTimeout	Indicates the idle timeout interval for executors with cached blocks.	• JDBCServer2x: 2147483647s • IndexServer2x: 2147483647s • SparkResource2x: 120

Spark Streaming

Spark Streaming is a streaming data processing function provided by the Spark batch processing platform. It processes data input from external systems in **mini-batch** mode.

Configure the following parameters in the **spark-defaults.conf** file on the Spark client.

Table 24-43 Parameter description

Parameter	Description	Default Value
spark.streaming.receiver.writeAheadLog.enable	Indicates whether to enable the write-ahead log (WAL) function. After this function is enabled, all input data received by the receiver is saved in the WAL. WAL ensures that data can be restored if the driver program becomes faulty.	false
spark.streaming.unpersist	Determines whether to automatically remove RDDs generated and saved by Spark Streaming from the Spark memory. If this function is enabled, original data received by Spark Streaming is also automatically cleared. If this function is disabled, original data and RDDs cannot be automatically cleared. External applications can access the data in Streaming. This, however, occupies more Spark memory resources.	true

Spark Streaming Kafka

The receiver is an important component of Spark Streaming. It receives external data, encapsulates the data into blocks, and provides the blocks for Streaming to

consume. The most common data source is Kafka. Spark Streaming integrates Kafka to ensure reliability and can directly use Kafka as the RDD input.

Table 24-44 Parameter description

Parameter	Description	Default Value
spark.streaming.kafka.maxRatePerPartition	Indicates the maximum rate (number of records per second) for reading data from each Kafka partition if the Kafka direct stream API is used.	-
spark.streaming.blockInterval	Indicates the interval (ms) for accumulating data received by a Spark Streaming receiver into a data block before the data is stored in Spark. A minimum value of 50 ms is recommended.	200ms
spark.streaming.receiver.maxRate	Indicates the maximum rate (number of records per second) for each receiver to receive data. The value **0** or a negative value indicates no limit to the rate.	-
spark.streaming.receiver.writeAheadLog.enable	Indicates whether to use ReliableKafkaReceiver. This receiver ensures the integrity of streaming data.	false

Netty/NIO and Hash/Sort Configuration

Shuffle is critical for big data processing, and the network is critical for the entire shuffle process. Currently, Spark supports two shuffle modes: hash and sort. There are two network modes: Netty and NIO.

Table 24-45 Parameter description

Parameter	Description	Default Value
spark.shuffle.manager	Indicates the data processing mode. There are two implementation modes: sort and hash. The sort shuffle has a higher memory utilization. It is the default option in Spark 1.2 and later versions. Spark 2.*x* and later versions do not support hash.	SORT

Parameter	Description	Default Value
spark.shuffle.consolidateFiles	(Only in hash mode) To merge intermediate files created during shuffle, set this parameter to **true**. Decreasing the number of files to be created can improve the processing performance of the file system and reduce risks. If the **ext4** or **xfs** file system is used, you are advised to set this parameter to **true**. Due to file system restrictions, this setting on **ext3** may reduce the processing performance of a server with more than eight cores.	false
spark.shuffle.sort.bypassMergeThreshold	This parameter is valid only when **spark.shuffle.manager** is set to **sort**. When Map aggregation is not performed and the number of partitions for Reduce tasks is less than or equal to the value of this parameter, do not merge and sort data to prevent performance deterioration caused by unnecessary sorting.	200
spark.shuffle.io.maxRetries	(Only in Netty mode) If this parameter is set to a non-zero value, fetch failures caused by I/O-related exceptions will be automatically retried. This retry logic helps the large shuffle keep stable when long GC pauses or intermittent network disconnections occur.	12
spark.shuffle.io.numConnectionsPerPeer	(Only in Netty mode) Connections between hosts are reused to reduce the number of connections between large clusters. For a cluster with many disks but a few hosts, this function may make concurrent requests unable to occupy all disks. Therefore, you can increase the value of this parameter.	1
spark.shuffle.io.preferDirectBufs	(Only in Netty mode) The off-heap buffer is used to reduce GC during shuffle and cache block transfer. In an environment where off-heap memory is strictly limited, you can disable it to force all applications from Netty to use heap memory.	true
spark.shuffle.io.retryWait	(Only in Netty mode) Specifies the duration for waiting for fetch retry, in seconds. The maximum delay caused by retry is **maxRetries** x **retryWait**. The default value is 15 seconds.	5

Common Shuffle Configuration

Table 24-46 Parameter description

Parameter	Description	Default Value
spark.shuffle.spill	If this parameter is set to **true**, data is overflowed to the disk to limit the memory usage during a reduce task.	true
spark.shuffle.spill.compress	Indicates whether to compress the data overflowed during shuffle. The algorithm specified by **spark.io.compression.codec** is used for data compression.	true
spark.shuffle.file.buffer	Specifies the size of the memory buffer for storing output streams of each shuffle file, in KB. These buffers can reduce the number of disk seek and system calls during the creation of intermediate shuffle file streams. You can also set this parameter by setting **spark.shuffle.file.buffer.kb**.	32KB
spark.shuffle.compress	Indicates whether to compress the output files of a Map task. You are advised to compress the broadcast variables. using **spark.io.compression.codec**.	true
spark.reducer.maxSizeInFlight	Specifies the maximum output size of the Map task that fetches data from each Reduce task, in MB. Each output requires a buffer, which is the fixed memory overhead of each Reduce task. Therefore, keep the value small unless there is a large amount of memory. You can also set this parameter by setting **spark.reducer.maxMbInFlight**.	48MB

Driver Configuration

Spark driver can be considered as the client of Spark applications. All code parsing is completed in this process. Therefore, the parameters of this process are especially important. The following describes how to configure parameters for Spark driver.

- **JavaOptions**: parameter following **-D** in the Java command, which can be obtained by **System.getProperty**
- **ClassPath**: path for loading the Java classes and Native library
- **Java Memory and Cores**: memory and CPU usage of the Java process
- **Spark Configuration**: Spark internal parameter, which is irrelevant to the Java process

Table 24-47 Parameter description

Parameter	Description	Default Value
spark.driver.extraJavaOptions	Indicates a series of extra JVM options passed to the driver, for example, GC setting and logging. Note: In client mode, this configuration cannot be set directly in the application using SparkConf because the driver JVM has been started. You can use **--driver-java-options** or the default property file to set the parameter.	For details, see **Configuring Spark Parameters Rapidly**.
spark.driver.extraClassPath	Indicates the extra class path entries attached to the class path of the driver. Note: In client mode, this configuration cannot be set directly in the application using SparkConf because the driver JVM has been started. You can use **--driver-java-options** or the default property file to set the parameter.	For details, see **Configuring Spark Parameters Rapidly**.
spark.driver.userClassPathFirst	(Trial) Indicates whether to allow JAR files added by users to take precedence over Spark JAR files when classes are loaded in the driver. This feature can be used to mitigate conflicts between Spark dependencies and user dependencies. This feature is in the trial phase and is used only in cluster mode.	false
spark.driver.extraLibraryPath	Sets a special library path for starting the driver JVM. Note: In client mode, this configuration cannot be set directly in the application using SparkConf because the driver JVM has been started. You can use **--driver-java-options** or the default property file to set the parameter.	• JDBCServer2x: ${SPARK_INSTALL_HOME}/spark/native • SparkResource2x: ${DATA_NODE_INSTALL_HOME}/hadoop/lib/native

Parameter	Description	Default Value
spark.driver.cores	Specifies the number of cores used by the driver process. This parameter is available only in cluster mode.	1
spark.driver.memory	Indicates the memory used by the driver process, that is, the memory used by the SparkContext initialization process (for example, 512 MB and 2 GB). Note: In client mode, this configuration cannot be set directly in the application using SparkConf because the driver JVM has been started. You can use **--driver-java-options** or the default property file to set the parameter.	4G
spark.driver.maxResultSize	Indicates the total size of serialization results of all partitions for each Spark action operation (for example, collect). The value must be at least 1 MB. If this parameter is set to **0**, the size is not limited. If the total amount exceeds this limit, the task will be aborted. If the value is too large, the memory of the driver may be insufficient (depending on the object memory overhead of **spark.driver.memory** and JVM). Set a proper limit to ensure sufficient memory for the driver.	1G
spark.driver.host	Specifies the host name or IP address monitored by the driver, which is used for the driver to communicate with the executor.	(local hostname)
spark.driver.port	Specifies the port monitored by the driver, which is used for the driver to communicate with the executor.	(random)

ExecutorLauncher Configuration

ExecutorLauncher exists only in Yarn-client mode. In Yarn-client mode, ExecutorLauncher and the driver are not in the same process. Therefore, you need to configure parameters for ExecutorLauncher.

Table 24-48 Parameter description

Parameter	Description	Default Value
spark.yarn.am.extraJavaOptions	Indicates a string of extra JVM options to pass to the YARN ApplicationMaster in client mode. Use **spark.driver.extraJavaOptions** in cluster mode.	For details, see **Configuring Spark Parameters Rapidly**.
spark.yarn.am.memory	Indicates the amount of memory to use for the YARN ApplicationMaster in client mode, in the same format as JVM memory strings (for example, 512 MB or 2 GB). In cluster mode, use **spark.driver.memory** instead.	1G
spark.yarn.am.memoryOverhead	This parameter is the same as **spark.yarn.driver.memoryOverhead**. However, this parameter applies only to ApplicationMaster in client mode.	-
spark.yarn.am.cores	Indicates the number of cores to use for the YARN ApplicationMaster in client mode. Use **spark.driver.cores** in cluster mode.	1

Executor Configuration

An executor is a Java process. However, unlike the driver and ApplicationMaster, an executor can have multiple processes. Spark supports only same configurations. That is, the process parameters of all executors must be the same.

Table 24-49 Parameter description

Parameter	Description	Default Value
spark.executor.extraJavaOptions	Indicates extra JVM option passed to the executor, for example, GC setting and logging. Do not set Spark attributes or heap size using this option. Instead, set Spark attributes using the SparkConf object or the **spark-defaults.conf** file specified when the spark-submit script is called. Set heap size using **spark.executor.memory**.	For details, see **Configuring Spark Parameters Rapidly**.

Parameter	Description	Default Value
spark.executor.extraClassPath	Indicates the extra classpath attached to the executor classpath. This parameter ensures compatibility with historical versions of Spark. Generally, you do not need to set this parameter.	-
spark.executor.extraLibraryPath	Sets the special library path used when the executor JVM is started.	For details, see **Configuring Spark Parameters Rapidly**.
spark.executor.userClassPathFirst	(Trial) Same function as **spark.driver.userClassPathFirst**. However, this parameter applies to executor instances.	false
spark.executor.memory	Indicates the memory size used by each executor process. Its character sting is in the same format as the JVM memory (example: 512 MB or 2 GB).	4G
spark.executorEnv.[EnvironmentVariableName]	Adds the environment variable specified by **EnvironmentVariableName** to the executor process. You can specify multiple environment variables.	-
spark.executor.logs.rolling.maxRetainedFiles	Sets the number of latest log files to be retained by the system during rolling. The old log files are deleted. This function is disabled by default.	-
spark.executor.logs.rolling.size.maxBytes	Sets the maximum size of the executor log file for rolling. This function is disabled by default. The value is in bytes. To automatically clear old logs, see **spark.executor.logs.rolling.maxRetainedFiles**.	-

Parameter	Description	Default Value
spark.executor.logs.rolling.strategy	Sets the executor log rolling policy. Rolling is disabled by default. The value can be **time** (time-based rolling) or **size** (size-based rolling). If this parameter is set to **time**, the value of the **spark.executor.logs.rolling.time.interval** attribute is used as the log rolling interval. If this parameter is set to **size**, **spark.executor.logs.rolling.size.maxBytes** is used to set the maximum size of the file for rolling.	-
spark.executor.logs.rolling.time.interval	Sets the time interval for executor log rolling. This function is disabled by default. The value can be **daily**, **hourly**, **minutely**, or any number of seconds. To automatically clear old logs, see **spark.executor.logs.rolling.maxRetainedFiles**.	daily

WebUI

The Web UI displays the running process and status of the Spark application.

Table 24-50 Parameter description

Parameter	Description	Default Value
spark.ui.killEnabled	Allows stages and jobs to be stopped on the web UI. **NOTE** For security purposes, the default value of this parameter is set to **false** to prevent misoperations. To enable this function, set this parameter to **true** in the **spark-defaults.conf** configuration file. Exercise caution when performing this operation.	true

Parameter	Description	Default Value
spark.ui.port	Specifies the port for your application's dashboard, which displays memory and workload data.	• JDBCServer2x: **4040** • SparkResource2x: 0 • IndexServer2x: 22901
spark.ui.retainedJobs	Specifies the number of jobs recorded by the Spark UI and status API before GC.	1000
spark.ui.retainedStages	Specifies the number of stages recorded by the Spark UI and status API before GC.	1000

HistoryServer

A History Server reads the **EventLog** file in the file system and displays the running status of the Spark application.

Table 24-51 Parameter description

Parameter	Description	Default Value
spark.history.fs.logDirectory	Specifies the log directory of a History Server.	-
spark.history.ui.port	Specifies the port for JobHistory listening to connection.	18080
spark.history.fs.updateInterval	Specifies the update interval of the information displayed on a History Server, in seconds. Each update checks for changes made to the event logs in the persistent store.	10s
spark.history.fs.update.interval.seconds	Specifies the interval for checking the update of each event log. This parameter has the same function as **spark.history.fs.updateInterval**. **spark.history.fs.updateInterval** is recommended.	10s

Parameter	Description	Default Value
spark.history.updateInterval	This parameter has the same function as **spark.history.fs.update.interval.seconds** and **spark.history.fs.updateInterval**. **spark.history.fs.updateInterval** is recommended.	10s

History Server UI Timeout and Maximum Number of Access Times

Table 24-52 Parameter description

Parameter	Description	Default Value
spark.session.maxAge	Specifies the session timeout interval, in seconds. This parameter applies only to the security mode. This parameter cannot be set in normal mode.	600
spark.connection.maxRequest	Specifies the maximum number of concurrent client access requests to JobHistory.	5000

EventLog

During the running of Spark applications, the running status is written into the file system in JSON format in real time for the History Server service to read and reproduce the application running status.

Table 24-53 Parameter description

Parameter	Description	Default Value
spark.eventLog.enabled	Indicates whether to log Spark events, which are used to reconstruct the web UI after the application execution is complete.	true
spark.eventLog.dir	Indicates the directory for logging Spark events if **spark.eventLog.enabled** is set to **true**. In this directory, Spark creates a subdirectory for each application and logs events of the application in the subdirectory. You can also set a unified address similar to the HDFS directory so that the History Server can read historical files.	hdfs://hacluster/spark2xJobHistory2x

Parameter	Description	Default Value
spark.eventLog.compress	Indicates whether to compress logged events when **spark.eventLog.enabled** is set to **true**.	false

Periodic Clearing of Event Logs

Event logs on JobHistory increases with submitted tasks. Too many event log files exist as the number of submitted tasks increases. Spark provides the function for periodically clearing event logs. You can enable this function and set the clearing interval using related parameters.

Table 24-54 Parameter description

Parameter	Description	Default Value
spark.history.fs.cleaner.enabled	Indicates whether to enable the clearing function.	true
spark.history.fs.cleaner.interval	Indicates the check interval of the clearing function.	1d
spark.history.fs.cleaner.maxAge	Indicates the maximum duration for storing logs.	4d

Kryo

Kryo is a highly efficient Java serialization framework, which is integrated into Spark by default. Almost all Spark performance tuning requires the process of converting the default serializer of Spark into a Kryo serializer. Kryo serialization supports only serialization at the Spark data layer. To configure Kryo serialization, set **spark.serializer** to **org.apache.spark.serializer.KryoSerializer** and configure the following parameters to optimize Kryo serialization performance:

Table 24-55 Parameter description

Parameter	Description	Default Value
spark.kryo.classesToRegister	Specifies the name of the class that needs to be registered with Kryo when Kryo serialization is used. Multiple classes are separated by commas (,).	-

Parameter	Description	Default Value
spark.kryo.referenceTracking	Indicates whether to trace the references to the same object when Kryo is used to serialize data. This function is applicable to the scenario where the object graph has circular references or the same object has multiple copies. Otherwise, you can disable this function to improve performance.	true
spark.kryo.registrationRequired	Indicates whether Kryo is used to register an object. When this parameter is set to **true**, an exception is thrown if an object that is not registered with Kryo is serialized. When it is set to **false** (default value), Kryo writes unregistered class names to the serialized object. This operation causes a large amount of performance overhead. Therefore, you need to enable this option before deleting a class from the registration queue.	false
spark.kryo.registrator	If Kryo serialization is used, use Kryo to register the class with the custom class. Use this property if you need to register a class in a custom way, such as specifying a custom field serializer. Otherwise, use **spark.kryo.classesToRegister**, which is simpler. Set this parameter to a class that extends KryoRegistrator.	-
spark.kryoserializer.buffer.max	Specifies the maximum size of the Kryo serialization buffer, in MB. The value must be greater than the object that attempts to be serialized. If the error "buffer limit exceeded" occurs in Kryo, increase the value of this parameter. You can also set this parameter by setting **spark.kryoserializer.buffer.max**.	64MB
spark.kryoserializer.buffer	Specifies the initial size of the Kryo serialization buffer, in MB. Each core of each worker has a buffer. If necessary, the buffer size will be increased to the value of **spark.kryoserializer.buffer.max**. You can also set this parameter by setting **spark.kryoserializer.buffer**.	64KB

Broadcast

Broadcast is used to transmit data blocks between Spark processes. In Spark, broadcast can be used for JAR packages, files, closures, and returned results. Broadcast supports two modes: Torrent and HTTP. The Torrent mode divides data into small fragments and distributes them to clusters. Data can be obtained

remotely if necessary. The HTTP mode saves files to the local disk and transfers the entire files to the remote end through HTTP if necessary. The former is more stable than the latter. Therefore, Torrent is the default broadcast mode.

Table 24-56 Parameter description

Parameter	Description	Default Value
spark.broadcast.factory	Indicates the broadcast mode.	org.apache.spark.broadcast.TorrentBroadcastFactory
spark.broadcast.blockSize	Indicates the block size of **TorrentBroadcastFactory**. If the value is too large, the concurrency during broadcast is reduced (the speed is slow). If the value is too small, BlockManager performance may be affected.	4096
spark.broadcast.compress	Indicates whether to compress broadcast variables before sending them. You are advised to compress the broadcast variables.	true

Storage

Spark features in-memory computing. Spark Storage is used to manage memory resources. Storage stores data blocks generated during RDD caching. The heap memory in the JVM acts as a whole. Therefore, **Storage Memory Size** is an important concept during Spark Storage management.

Table 24-57 Parameter description

Parameter	Description	Default Value
spark.storage.memoryMapThreshold	Specifies the block size. If the size of a block exceeds the value of this parameter, Spark performs memory mapping for the disk file. This prevents Spark from mapping too small blocks during memory mapping. Generally, memory mapping for blocks whose page size is close to or less than that of the operating system has high overhead.	2m

PORT

Table 24-58 Parameter description

Parameter	Description	Default Value
spark.ui.port	Specifies the port for your application's dashboard, which displays memory and workload data.	• JDBCServer2x: **4040** • SparkResource2x: 0
spark.blockManager.port	Specifies all ports monitored by BlockManager. These ports are on both the driver and executor.	**Range of Random Ports**
spark.driver.port	Specifies the port monitored by the driver, which is used for the driver to communicate with the executor.	**Range of Random Ports**

Range of Random Ports

All random ports must be within a certain range.

Table 24-59 Parameter description

Parameter	Description	Default Value
spark.random.port.min	Sets the minimum random port.	22600
spark.random.port.max	Sets the maximum random port.	22899

TIMEOUT

By default, computation tasks that can well process medium-scale data are configured in Spark. However, if the data volume is too large, the tasks may fail due to timeout. In the scenario with a large amount of data, the timeout parameter in Spark needs to be assigned a larger value.

Table 24-60 Parameter description

Parameter	Description	Default Value
spark.files.fetchTimeout	Specifies the communication timeout (in seconds) when fetching files added using **SparkContext.addFile()** of the driver.	60s
spark.network.timeout	Specifies the default timeout for all network interactions, in seconds. You can use this parameter to replace **spark.core.connection.ack.wait.timeout**, **spark.akka.timeout**, **spark.storage.blockManagerSlaveTimeoutMs**, or **spark.shuffle.io.connectionTimeout**.	360s
spark.core.connection.ack.wait.timeout	Specifies the timeout for a connection to wait for a response, in seconds. To avoid long-time waiting caused by GC, you can set this parameter to a larger value.	60

Encryption

Spark supports SSL for Akka and HTTP (for the broadcast and file server) protocols, but does not support SSL for the web UI and block transfer service.

SSL must be configured on each node and configured for each component involved in communication using a particular protocol.

Table 24-61 Parameter description

Parameter	Description	Default Value
spark.ssl.enabled	Indicates whether to enable SSL connections for all supported protocols. All SSL settings similar to **spark.ssl.*xxx*** indicate the global configuration of all supported protocols. To override the global configuration of a particular protocol, you must override the property in the namespace specified by the protocol. Use **spark.ssl.YYY.XXX** to overwrite the global configuration of the particular protocol specified by **YYY**. **YYY** can be either **akka** for Akka-based connections or **fs** for the broadcast and file server.	false
spark.ssl.enabledAlgorithms	Indicates the comma-separated list of passwords. The specified passwords must be supported by the JVM.	-
spark.ssl.keyPassword	Specifies the password of a private key in the keystore.	-

Parameter	Description	Default Value
spark.ssl.keyStore	Specifies the path of the keystore file. The path can be absolute or relative to the directory where the component is started.	-
spark.ssl.keyStorePassword	Specifies the password of the keystore.	-
spark.ssl.protocol	Specifies the protocol name. This protocol must be supported by the JVM. The reference list of protocols is available on this page.	-
spark.ssl.trustStore	Specifies the path of the truststore file. The path can be absolute or relative to the directory where the component is started.	-
spark.ssl.trustStorePassword	Specifies the password of the truststore.	-

Security

Spark supports shared key-based authentication. You can use **spark.authenticate** to configure authentication. This parameter controls whether the Spark communication protocol uses the shared key for authentication. This authentication is a basic handshake that ensures that both sides have the same shared key and are allowed to communicate. If the shared keys are different, the communication is not allowed. You can create shared keys as follows:

- For Spark on YARN deployments, set **spark.authenticate** to **true**. Then, shared keys are automatically generated and distributed. Each application exclusively occupies a shared key.
- For other types of Spark deployments, configure Spark parameter **spark.authenticate.secret** on each node. All masters, workers, and applications use this key.

Table 24-62 Parameter description

Parameter	Description	Default Value
spark.acls.enable	Indicates whether to enable Spark ACLs. If Spark ACLs are enabled, the system checks whether the user has the permission to access and modify jobs. Note that this requires the user to be identifiable. If the user is identified as invalid, the check will not be performed. Filters can be used to verify and set users on the UI.	true

Parameter	Description	Default Value
spark.admin.acls	Specifies the comma-separated list of users/administrators that have the permissions to view and modify all Spark jobs. This list can be used if you are running on a shared cluster and working with the help of an MRS cluster administrator or developer.	admin
spark.authenticate	Indicates whether Spark authenticates its internal connections. If the application is not running on YARN, see **spark.authenticate.secret**.	true
spark.authenticate.secret	Sets the key for authentication between Spark components. This parameter must be set if Spark does not run on YARN and authentication is disabled.	-
spark.modify.acls	Specifies the comma-separated list of users who have the permission to modify Spark jobs. By default, only users who have enabled Spark jobs have the permission to modify the list (for example, delete the list).	-
spark.ui.view.acls	Specifies the comma-separated list of users who have the permission to access the Spark web UI. By default, only users who have enabled Spark jobs have the access permission.	-

Enabling the Authentication Mechanism Between Spark Processes

Spark currently supports authentication via a shared secret. You can determine whether to enable Spark authentication during communication by configuring **spark.authenticate**. In this mode of authentication, both communicating parties utilize a shared secret, established through a straightforward handshake process.

Configure the following parameters in the **spark-defaults.conf** file on the Spark client.

Table 24-63 Parameter description

Parameter	Description	Default Value
spark.authenticate	For Spark on YARN deployments, set this parameter to **true**. Then, keys are automatically generated and distributed, and each application uses a unique key.	true

Compression

Data compression is policy that optimizes memory usage at the expense of CPU. Therefore, when the Spark memory is severely insufficient (this issue is common due to the characteristics of in-memory computing), data compression can greatly improve performance. Spark supports three types of compression algorithm: Snappy, LZ4, and LZF. Snappy is the default compression algorithm and invokes the native method to compress and decompress data. In YARN mode, pay attention to the impact of non-heap memory on the container process.

Table 24-64 Parameter description

Parameter	Description	Default Value
spark.io.compression.codec	Indicates the codec for compressing internal data, such as RDD partitions, broadcast variables, and shuffle output. By default, Spark supports three types of compression algorithm: LZ4, LZF, and Snappy. You can specify algorithms using fully qualified class names, such as **org.apache.spark.io.LZ4CompressionCodec**, **org.apache.spark.io.LZFCompressionCodec**, and **org.apache.spark.io.SnappyCompressionCodec**.	lz4
spark.io.compression.lz4.block.size	Indicates the block size (bytes) used in LZ4 compression when the LZ4 compression algorithm is used. When LZ4 is used, reducing the block size also reduces the shuffle memory usage.	32768
spark.io.compression.snappy.block.size	Indicates the block size (bytes) used in Snappy compression when the Snappy compression algorithm is used. When Snappy is used, reducing the block size also reduces the shuffle memory usage.	32768
spark.shuffle.compress	Indicates whether to compress the output files of a Map task. You are advised to compress the broadcast variables. using **spark.io.compression.codec**.	true
spark.shuffle.spill.compress	Indicates whether to compress the data overflowed during shuffle using **spark.io.compression.codec**.	true
spark.eventLog.compress	Indicates whether to compress logged events when **spark.eventLog.enabled** is set to **true**.	false
spark.broadcast.compress	Indicates whether to compress broadcast variables before sending them. You are advised to compress the broadcast variables.	true

Parameter	Description	Default Value
spark.rdd.compress	Indicates whether to compress serialized RDD partitions (for example, the **StorageLevel.MEMORY_ONLY_SER** partition). Substantial space can be saved at the cost of some extra CPU time.	false

Reducing the Probability of Abnormal Client Application Operations When Resources Are Insufficient

When resources are insufficient, ApplicationMaster tasks must wait and will not be processed until enough resources are available for use. If the actual waiting time exceeds the configured waiting time, the ApplicationMaster tasks will be deleted. Adjust the following parameters to reduce the probability of abnormal client application operation.

Configure the following parameters in the **spark-defaults.conf** file on the client.

Table 24-65 Parameter description

Parameter	Description	Default Value
spark.yarn.applicationMaster.waitTries	Specifies the number of the times that ApplicationMaster waits for Spark master, which is also the times that ApplicationMaster waits for SparkContext initialization. Enlarge this parameter value to prevent ApplicationMaster tasks from being deleted and reduce the probability of abnormal client application operations.	10
spark.yarn.am.memory	Specifies the ApplicationMaster memory. Enlarge this parameter value to prevent ApplicationMaster tasks from being deleted by ResourceManager due to insufficient memory and reduce the probability of abnormal client application operations.	1G

24.15.3 Spark Log Overview

Log Description

Log paths:

- Executor run log: **${BIGDATA_DATA_HOME}/hadoop/data${i}/nm/containerlogs/application_${appid}/container_{$contid}**

NOTE

The logs of running tasks are stored in the preceding path. After the running is complete, the system determines whether to aggregate the logs to an HDFS directory based on the YARN configuration. For details, see **YARN Common Configuration Parameters**.

- Other logs: **/var/log/Bigdata/spark2x**

Log archiving rule:

- When tasks are submitted in **yarn-client** or **yarn-cluster** mode, executor log files are stored each time when the size of the log files reaches 50 MB. A maximum of 10 log files can be reserved without being compressed.
- The JobHistory2x log file is backed up each time when the size of the log file reaches 100 MB. A maximum of 100 log files can be reserved without being compressed.
- The JDBCServer2x log file is backed up each time when the size of the log file reaches 100 MB. A maximum of 100 log files can be reserved without being compressed.
- The IndexServer2x log file is backed up each time when the size of the log file reaches 100 MB. A maximum of 100 log files can be reserved without being compressed.
- The JDBCServer2x audit log file is backed up each time when the size of the log file reaches 20 MB by default. A maximum of 20 log files can be reserved without being compressed.
- The log file size and the number of compressed files to be reserved can be configured on FusionInsight Manager.

Table 24-66 Spark2x log list

Log Type	Name	Description
SparkResource2x logs	spark.log	Spark2x service initialization log
	prestart.log	Prestart script log
	cleanup.log	Cleanup log file for instance installation and uninstallation
	spark-availability-check.log	Spark2x service health check log
	spark-service-check.log	Spark2x service check log
JDBCServer2x logs	JDBCServer-start.log	JDBCServer2x startup log
	JDBCServer-stop.log	JDBCServer2x stop log
	JDBCServer.log	JDBCServer2x run log on the server
	jdbc-state-check.log	JDBCServer2x health check log

Log Type	Name	Description
	jdbcserver-omm-pid***-gc.log.*.current	IJDBCServer2x process GC log
	spark-omm-org.apache.spark.sql.hive.thriftserver.HiveThriftProxyServer2-***.out*	JDBCServer2x process startup log. If the process stops, the **jstack** information is printed.
JobHistory2x logs	jobHistory-start.log	JobHistory2x startup log
	jobHistory-stop.log	JobHistory2x stop log
	JobHistory.log	JobHistory2x running process log
	jobhistory-omm-pid***-gc.log.*.current	JobHistory2x process GC log
	spark-omm-org.apache.spark.deploy.history.HistoryServer-***.out*	JobHistory2x process startup log. If the process stops, the **jstack** information is printed.
IndexServer2x logs	IndexServer-start.log	IndexServer2x startup log
	IndexServer-stop.log	IndexServer2x stop log
	IndexServer.log	IndexServer2x run log on the server
	indexserver-state-check.log	IndexServer2x health check log
	indexserver-omm-pid***-gc.log.*.current	IndexServer2x process GC log
	spark-omm-org.apache.spark.sql.hive.thriftserver.IndexServerProxy-***.out*	IndexServer2x process startup log. If the process stops, the **jstack** information is printed.
Audit Log	jdbcserver-audit.log ranger-audit.log	JDBCServer2x audit log

Log levels

Table 24-67 describes the log levels supported by Spark2x. The priorities of log levels are ERROR, WARN, INFO, and DEBUG in descending order. Logs whose levels are higher than or equal to the specified level are printed. The number of printed logs decreases as the specified log level increases.

Table 24-67 Log levels

Level	Description
ERROR	Error information about the current event processing
WARN	Exception information about the current event processing
INFO	Logs of this level record normal running status information about the system and events.
DEBUG	Logs of this level record the system information and system debugging information.

To modify log levels, perform the following operations:

NOTE

By default, the service does not need to be restarted after the Spark2x log levels are configured.

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **Spark2x** and click **Configurations**.

Step 3 Select **All Configurations**.

Step 4 On the menu bar on the left, select the log menu of the target role.

Step 5 Select a desired log level.

Step 6 Click **Save**. Then, click **OK**.

----End

Log Format

Table 24-68 Log Format

Type	Format	Example
Run log	<*yyyy-MM-dd HH:mm:ss,SSS*>|<*Log level*>|<*Name of the thread that generates the log*>|<*Message in the log*>|<*Location where the log event occurs*>	2014-09-22 11:16:23,980 INFO DAGScheduler: Final stage: Stage 0(reduce at SparkPi.scala:35)

24.15.4 Obtaining Container Logs of a Running Spark Application

Container logs of running Spark applications are distributed on multiple nodes. This section describes how to quickly obtain container logs.

Scenario Description

You can run the **yarn logs** command to obtain the logs of applications running on YARN. In different scenarios, you can run the following commands to obtain required logs:

1. Obtain complete logs of the application: **yarn logs --applicationId** *<appId>* **-out** *<outputDir>*.

   Example: **yarn logs --applicationId application_1574856994802_0016 -out /opt/test**

   The following figure shows the command output.

   a. If the application is running, container logs in the **dead** state cannot be obtained.

   b. If the application is stopped, all archived container logs can be obtained.

2. Obtain logs of a specified container: **yarn logs -applicationId** *<appId>* **-containerId** *<containerId>*.

   Example: **yarn logs -applicationId application_1574856994802_0018 -containerId container_e01_1574856994802_0018_01_000003**

   The following figure shows the command output.

   a. If the application is running, container logs in the **dead** state cannot be obtained.

   b. If the application is stopped, you can obtain logs of any container.

3. Obtain container logs in any state: **yarn logs -applicationId** *<appId>* **-containerId** *<containerId>* **-nodeAddress** *<nodeAddress>*

   Example: **yarn logs -applicationId application_1574856994802_0019 -containerId container_e01_1574856994802_0019_01_000003 -nodeAddress 192-168-1-1:**8041

   Execution result: Logs of any container can be obtained.

   NOTE

   You need to set *nodeAddress* in the command. You can run the following command to obtain the value:

   yarn node -list -all

24.15.5 Changing Spark Log Levels

Scenarios

In some scenarios, to locate problems or check information by changing the log level,

you can add the **-Dlog4j.configuration.watch=true** parameter to the JVM parameter of a process before the process is started. After the process is started,

you can modify the log4j configuration file corresponding to the process to change the log level.

The following processes support the dynamic setting of log levels: driver, executor, ApplicationMaster, JobHistory and JDBCServer.

Allowed log levels are as follows: FATAL, ERROR, WARN, INFO, DEBUG, TRACE, and ALL.

Configuration Description

Add the following parameters to the JVM parameter corresponding to a process.

Table 24-69 Parameter description

Parameter	Description	Default Value
-Dlog4j.configuration.watch	Indicates a JVM parameter of a process. If this parameter is set to **true**, the dynamic configuration of log levels is enabled.	Left blank, indicating that the dynamic configuration of log levels is disabled

Table 24-70 lists the JVM parameters of the driver, executor, and ApplicationMaster processes. Configure the following parameters in the **spark-defaults.conf** file on the Spark client. Set the log levels of the driver, executor, and ApplicationMaster processes in the log4j configuration file specified by the **-Dlog4j.configuration** parameter.

Table 24-70 JVM parameters of processes (1)

Parameter	Description	Default Log Level
spark.driver.extraJavaOptions	Indicates the JVM parameter of the driver process.	INFO
spark.executor.extraJavaOptions	Indicates the JVM parameter of the executor process.	INFO
spark.yarn.am.extraJavaOptions	Indicates the JVM parameter of the ApplicationMaster process.	INFO

Table 24-71 describes the JVM parameters of JobHistory Server and JDBCServer. Set the parameters in the **ENV_VARS** configuration file. Set the log levels of JobHistory Server and JDBCServer in the **log4j.properties** configuration file.

Table 24-71 JVM parameters of processes (2)

Parameter	Description	Default Log Level
GC_OPTS	Indicates the JVM parameter of the JobHistory Server process.	INFO
SPARK_SUBMIT_OPTS	Indicates the JVM parameter of JDBCServer.	INFO

Example:

To change the log level of the executor process to DEBUG dynamically, modify the **spark.executor.extraJavaOptions** JVM parameter of the executor process in the **spark-defaults.conf** file and run the following command to add the following configuration before the process is started:

```
-Dlog4j.configuration.watch=true
```

After the user application is submitted, change the log level in the Log4j configuration file (for example, **-Dlog4j.configuration=file:${BIGDATA_HOME}/FusionInsight_Spark2x_xxx/install/FusionInsight-Spark2x-*/spark/conf/log4j-executor.properties**) specified by the **-Dlog4j.configuration** parameter in **spark.executor.extraJavaOptions** to **DEBUG**:

```
log4j.rootCategory=DEBUG, sparklog
```

It takes several seconds for the DEBUG level to take effect.

24.15.6 Viewing Container Logs on the Web UI

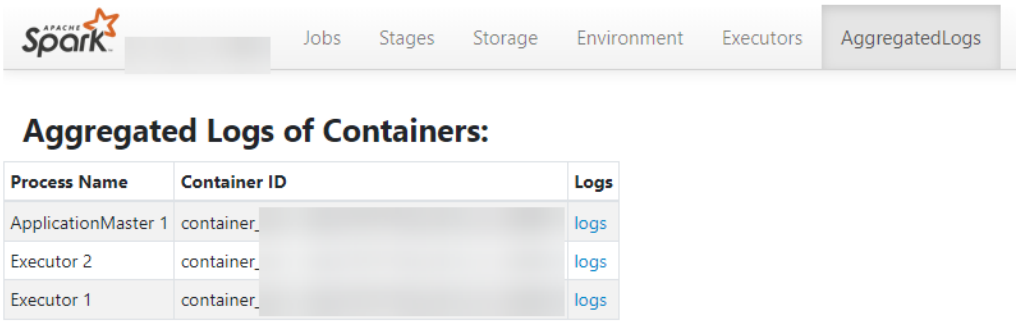
Scenarios

When **yarn.log-aggregation-enable** of YARN is set to **true**, the container log aggregation function is enabled. Log aggregation indicates that after applications are run on YARN, NodeManager aggregates all container logs of the node to HDFS and deletes local logs. For details, see **Configuring Container Log Aggregation**.

However, all logs will be aggregated to an HDFS directory and can only be viewed by accessing an HDFS file. Open-source Spark and YARN do not support the function of viewing aggregated logs on the web UI.

Spark supports this function. As shown in **Figure 24-9**, the **AggregatedLogs** tab is added to the HistoryServer page. You can click **logs** to view aggregated logs.

Figure 24-9 Aggregated log page



Configuration Description

To display logs on the web UI, aggregated logs need to be parsed and presented. Spark parses aggregation logs using JobHistoryServer of Hadoop. Therefore, you can use the **spark.jobhistory.address** parameter to specify the URL of the JobHistoryServer page to parse and present the logs.

Navigation path for setting parameters:

When submitting an application, set these parameters using **--conf** or adjust the following parameter in the **spark-defaults.conf** configuration file on the client.

NOTE

- This function depends on JobHistoryServer of Hadoop. Therefore, ensure that JobHistoryServer is running properly before using the log aggregation function.
- If the parameter value is empty, the **AggregatedLogs** tab page still exists, but you cannot view logs by clicking **logs**.
- The aggregated container logs can be viewed only when the application is running and event log files of the application exist on HDFS.
- You can click the log link on the **Executors** page to view the logs of a running task. After the task completes, the logs are aggregated to HDFS, and the log link on the **Executors** page becomes invalid. In this case, you can click **logs** on the **AggregatedLogs** page to view the aggregated logs.

Table 24-72 Parameter description

Parameter	Description	Default Value
spark.jobhistory.address	URL of the JobHistoryServer page. The format is *http(s)://ip:port/jobhistory*. For example, **https://10.92.115.1:26014/jobhistory**. The default value is empty, indicating that container aggregation logs cannot be viewed on the web UI. Restart the service for the configuration to take effect.	-

24.15.7 Configuring the Number of Lost Executors Displayed on the Web UI

Scenarios

On the Spark web UI, the **Executor** page can display information about lost executors. Executors of JDBCServer long tasks are dynamically recycled. Therefore, the number of displayed lost executors must be configured to fit the **Executor** page.

Configuration Description

Configure the following parameters in the **spark-defaults.conf** file of the Spark client.

Table 24-73 Parameter description

Parameter	Description	Default Value
spark.ui.retainedDeadExecutors	Specifies the maximum number of lost executors displayed on the Spark web UI.	100

24.15.8 Configuring Local Disk Cache for JobHistory

Scenarios

JobHistory can use local disks to cache historical data of Spark applications to prevent large volumes of application data from being loaded to the JobHistory memory and reduce memory usage. In addition, the cached data can be reused to accelerate access to the same application.

Parameters

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark2x** > **Configurations**, click **All Configurations**, and search for the following parameters.

Parameter	Description	Default Value
spark.history.store.path	Local directory for JobHistory to cache historical data. If this parameter is configured, JobHistory caches historical application data in local disks instead of the memory.	${BIGDATA_HOME}/tmp/spark2x_JobHistory

Parameter	Description	Default Value
spark.history.store.maxDiskUsage	Maximum available space for JobHistory to caching data in local disks	10g

24.15.9 Configuring Spark Event Log Rollback

Scenario

When the event log mode is enabled for Spark, that is, **spark.eventLog.enabled** is set to **true**, events are written to a configured log file to record the program running process. If a program, for example JDBCServer or Spark Streaming, runs for a long period of time and has run many jobs and tasks during this period, many events are recorded in the log file, significantly increasing the file size.

When log rollover is enabled, metadata events are written into the log file and job events are written into a new log file (whether a job event is written to the new log file depends on the file size). Metadata events include EnviromentUpdate, BlockManagerAdded, BlockManagerRemoved, UnpersistRDD, ExecutorAdded, ExecutorRemoved, MetricsUpdate, ApplicationStart, ApplicationEnd, and LogStart. Job events include StageSubmitted, StageCompleted, TaskResubmit, TaskStart, TaskEnd, TaskGettingResult, JobStart, and JobEnd. For Spark SQL applications, job events also include ExecutionStart and ExecutionEnd.

The UI for the HistoryServer service of Spark is obtained by reading and parsing these log files. The memory size is preset before the HistoryServer process starts. Therefore, when the size of log files is large, loading and parsing these files may cause problems such as insufficient memory and driver GC.

To load large log files in small memory mode, you need to enable log rollover for large applications. Generally, it is recommended that this function be enabled for long-running applications.

Parameters

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark2x** > **Configurations**, click **All Configurations**, and search for the following parameters.

Parameter	Description	Default Value
spark.eventLog.rolling.enabled	Whether to enable rollover for event log files. If this parameter is set to **true**, the size of each event log file is reduced to the configured size.	true

Parameter	Description	Default Value
spark.eventLog.rolling.maxFileSize	Maximum size of the event log file to be rolled over when **spark.eventlog.rolling.enabled** is set to **true**.	128M
spark.eventLog.compression.codec	Codec used to compress event logs. By default, Spark provides four types of codecs: LZ4, LZF, Snappy, and ZSTD. If this parameter is not specified, **spark.io.compression.codec** is used.	None
spark.eventLog.logStageExecutorMetrics	Whether to write each stage peak value (for each executor) of executor metrics to the event log.	false

24.15.10 Enhancing Stability in a Limited Memory Condition

Scenario

A large amount of memory is required when Spark SQL executes a query, especially during Aggregate and Join operations. If the memory is limited, OutOfMemoryError may occur. Stability in a limited memory condition ensures queries to be run in limited memory without OutOfMemoryError.

NOTE

Limited memory does not mean infinitely small memory, but ensures stable queries by using disks in a scenario where memory fails to store the data amount that is several times larger than the available memory size. For example, for queries involving Join, the data of the same key used for Join needs to be stored in memory. If the data amount is too large to be stored in the available memory, OutOfMemoryError occurs.

Stability in a limited memory condition involves the following sub-functions:

1. ExternalSort

   If the memory is inadequate during sorting, partial data overflows to disks.

2. TungstenAggregate

   By default, ExternalSort is used to sort data before data aggregation. Therefore, if the memory is inadequate, the data overflows to disks during sorting. The data has been properly sorted before aggregation and only aggregation results of the current key are remained, which use a small amount of memory.

3. SortMergeJoin and SortMergeOuterJoin

   SortMergeJoin and SortMergeOuterJoinan are based on the equivalence join of sorted data. By default, ExternalSort is used to sort the data before the equivalence join. Therefore, if the memory is inadequate, the data overflows to disks during sorting. The data has been properly sorted before the

equivalence join and only the data of the same key are remained, which uses a small amount of memory.

Configuration

Navigation path for setting parameters:

When submitting an application, set the following parameters using **--conf** or adjust the parameters in the **spark-defaults.conf** configuration file on the client.

Table 24-74 Parameter description

Parameter	Description	Default Value
spark.sql.tungsten.enabled	Type: Boolean ● If the value is **true**, tungsten is enabled. That is, the logic plan is equivalent to the codegeneration function, and the physical plan uses the corresponding tungsten execution plan. ● If the value is **false**, tungsten is disabled.	true
spark.sql.codegen.wholeStage	Type: Boolean ● If the value is **true**, codegeneration is enabled. That is, for some specified queries, the logic plan code will be generated dynamically when running. ● If the value is **false**, codegeneration is disabled and the existing static code is used.	true

NOTE

1. To enable ExternalSort, you need to set **spark.sql.planner.externalSort** to **true** and **spark.sql.unsafe.enabled** to **false** or **spark.sql.codegen.wholeStage** to **false**.
2. To enable TungstenAggregate, use either of the following methods:

   Set **spark.sql.codegen.wholeStage** and **spark.sql.unsafe.enabled** to **true** in the configuration file or CLI.

   If neither **spark.sql.codegen.wholeStage** nor **spark.sql.unsafe.enabled** is **true** or either of them is **true**, **TungstenAggregate** is enabled as long as **spark.sql.tungsten.enabled** is set to **true**.

24.15.11 Configuring Environment Variables in Yarn-Client and Yarn-Cluster Modes

Scenario

Values of some configuration parameters of Spark client vary depending on its work mode (YARN-Client or YARN-Cluster). If you switch Spark client between different modes without first changing values of such configuration parameters, Spark client fails to submit jobs in the new mode.

To avoid this, configure parameters as described in **Table 24-75**.

- In Yarn-Cluster mode, use the new parameters (path and parameters of Spark server).
- In Yarn-Client mode, uses the original parameters.

  They are **spark.driver.extraClassPath**, **spark.driver.extraJavaOptions**, and **spark.driver.extraLibraryPath**.

NOTE

If you choose not to add the parameters in **Table 24-75**, Spark client can continue to operate well in either mode but the mode switch requires changes to some of its configuration parameters.

Configuration Parameters

Navigation path for setting parameters:

On Manager, choose **Cluster** > **Services** > **Spark2x**, click **Configurations**, click **All Configurations**, and enter a parameter name in the search box.

Table 24-75 Parameter description

Parameter	Description	Default Value
spark.yarn.cluster.driver.extraClassPath	Indicates the extraClassPath of the driver in Yarn-cluster mode. Set the parameter to the path and parameters of the server. The original parameter **spark.driver.extraClassPath** indicates the extraClassPath of Spark client. By using different parameters to separate the settings of Spark server from the settings of Spark client, you can switch Spark client to different modes without changing parameter values.	${BIGDATA_HOME}/common/runtime/security

Parameter	Description	Default Value
spark.yarn.cluster.driver.extraJavaOptions	Indicates the extraJavaOptions of Driver in Yarn-Cluster mode and is set to path and parameters of extraJavaOptions of Spark server. The original parameter **spark.driver.extraJavaOptions** indicates the path of extraJavaOptions of Spark client. By using different parameters to separate the settings of Spark server from the settings of Spark client, you can switch Spark client to different modes without changing parameter values.	-Xloggc:<LOG_DIR>/indexserver-%p-gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -Dlog4j.configuration=./__spark_conf__/__hadoop_conf__/log4j-executor.properties -Dlog4j.configuration.watch=true -Djava.security.auth.login.config=./__spark_conf__/__hadoop_conf__/jaas-zk.conf -Dzookeeper.server.principal=${ZOOKEEPER_SERVER_PRINCIPAL} -Djava.security.krb5.conf=./__spark_conf__/__hadoop_conf__/kdc.conf -Djetty.version=x.y.z -Dorg.xerial.snappy.tempdir=${BIGDATA_HOME}/tmp -Dcarbon.properties.filepath=./__spark_conf__/__hadoop_conf__/carbon.properties -Djdk.tls.ephemeralDHKeySize=2048 -Dspark.ssl.keyStore=./child.keystore #{java_stack_prefer}

24.15.12 Broaden Support for Hive Partition Pruning Predicate Pushdown

Scenario

In earlier versions, the predicate for pruning Hive table partitions is pushed down. Only comparison expressions between column names and integers or character strings can be pushed down. In version 2.3, pushdown of the null, in, and, or expressions are supported.

Parameters

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark2x** > **Configurations**, click **All Configurations**, and search for the following parameters.

Parameter	Description	Default Value	Value Range
spark.sql.hive.advancedPartitionPredicatePushdown.enabled	Specifies whether to broaden the support for Hive partition pruning predicate pushdown.	true	[true,false]

24.15.13 Configuring the Column Statistics Histogram for Higher CBO Accuracy

Scenarios

Typically, Spark SQL statements are optimized using heuristic optimization rules. Such rules are provided only based on the characteristics of the logical plan and the characteristics of the data (the execution cost of the operator) are not considered. Spark 2.2 introduces cost-based optimizer (CBO). CBO gathers statistics on tables and columns, and uses this information to estimate the number of output records and byte size for each operator based on the input dataset. These estimates determine the cost of executing each operator.

CBO adjusts the execution plan to minimize the end-to-end query time. The idea is as follows:

- Filter out irrelevant data as early as possible.
- Minimize the cost of each operator.

The CBO optimization process is divided into two steps:

1. Collect statistics.
2. Estimate the output data set of a specific operator based on the input dataset.

Table-level statistics include the number of records and the total size of table data files.

Column-level statistics include the number of unique values, maximum value, minimum value, number of null values, average length, maximum length, and histogram.

After the statistics are obtained, the execution cost of the operator can be estimated. Common operators include the Filter and Join operators.

Histogram is a type of column statistics. It can clearly describe the distribution of column data. The column data is distributed to a specified number of bins that are displayed in ascending order by size. The upper and lower limits of each bin are calculated. The amount of data in all bins is the same (a contour histogram). With

the detailed distribution of data, the cost estimation of each operator is more accurate and the optimization effect is better.

This feature can be enabled by using the following parameter.

spark.sql.statistics.histogram.enabled: specifies whether to enable the histogram function. The default value is **false**.

Parameters

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Spark2x** > **Configurations**, click **All Configurations**, and search for the following parameters.

Parameter	Description	Default Value	Value Range
spark.sql.cbo.enabled	Whether to enable CBO to estimate the statistics of the execution plan.	false	[true,false]
spark.sql.cbo.joinReorder.enabled	Whether to enable CBO connection reordering.	false	[true,false]
spark.sql.cbo.joinReorder.dp.threshold	Maximum number of join nodes allowed in the dynamic planning algorithm.	12	≥ 1
spark.sql.cbo.joinReorder.card.weight	Proportion of the dimension (number of rows) in the cost comparison of the reconnection execution plan: Number of rows x Proportion + File size x (1 – Proportion).	0.7	0-1
spark.sql.statistics.size.autoUpdate.enabled	Whether to enable the function of automatically updating the table size when the table data changes. If there are a large number of data files in a table, this operation consumes a lot of resources and slows down data operations.	false	[true,false]

Parameter	Description	Default Value	Value Range
spark.sql.statistics.histogram.enabled	After this function is enabled, a histogram is generated when column information is collected. Histograms can improve estimation accuracy, but collecting histogram information requires additional workload.	false	[true,false]
spark.sql.statistics.histogram.numBins	Number of slots in the generated histogram.	254	≥ 2
spark.sql.statistics.ndv.maxError	Maximum estimation error allowed by the HyperLogLog++ algorithm when column-level statistics are generated.	0.05	0-1
spark.sql.statistics.percentile.accuracy	Accuracy of percentile estimation when generating equal height histograms. A larger value indicates more accuracy. The estimated error value can be obtained using 1.0/Percentile estimation accuracy.	10000	≥ 1

NOTE

- A histogram takes effect in CBO only when the following conditions are met:
  - **spark.sql.statistics.histogram.enabled**: The default value is **false**. Change the value to **true** to enable the histogram function.
  - **spark.sql.cbo.enabled**: The default value is **false**. Change the value to **true** to enable CBO.
  - **spark.sql.cbo.joinReorder.enabled**: The default value is **false**. Change the value to **true** to enable connection reordering.
- If a client is used to submit tasks, the modification of **spark.sql.cbo.enabled**, **spark.sql.cbo.joinReorder.enabled**, **spark.sql.cbo.joinReorder.dp.threshold**, **spark.sql.cbo.joinReorder.card.weight**, **spark.sql.statistics.size.autoUpdate.enabled**, **spark.sql.statistics.histogram.enabled**, **spark.sql.statistics.histogram.numBins**, **spark.sql.statistics.ndv.maxError**, and **spark.sql.statistics.percentile.accuracy** takes effect only after the client is downloaded again.

24.15.14 Using CarbonData for First Query

Tool Overview

The first query of CarbonData is slow, which may cause a delay for nodes that have high requirements on real-time performance.

The tool provides the following functions:

- Preheat the tables that have high requirements on query delay for the first time.

Tool Usage

Download and install the client. Assume that the installation directory is **/opt/client**. Go to the **/opt/client/Spark2x/spark/bin** directory and run the **start-prequery.sh** script.

Configure **prequeryParams.properties** by referring to **Table 24-76**.

Table 24-76 Parameters

Parameter	Description	Example
spark.prequery.period.max.minute	Maximum preheating duration, in minutes.	60
spark.prequery.tables	Table name configuration, *database.table:int*. The table name supports the wildcard (*). **int** indicates the duration (unit: day) within which the table is updated before it is preheated.	default.test*:10
spark.prequery.maxThreads	Maximum number of concurrent threads during preheating	50
spark.prequery.sslEnable	The value is **true** in security mode and **false** in non-security mode.	true
spark.prequery.driver	IP address and port number of JDBCServer. The format is *IP address:Port number*. If multiple servers need to be preheated, enter multiple *IP address:Port number* of the servers and separate them with commas (,).	192.168.0.2:22550

Parameter	Description	Example
spark.prequery.sql	SQL statement for preheating. Different statements are separated by colons (:).	SELECT COUNT(*) FROM %s;SELECT * FROM %s LIMIT 1
spark.security.url	URL required by JDBC in security mode	;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.hadoop.com@HADOOP.COM;

NOTE

The statement configured in **spark.prequery.sql** is executed in each preheated table. The table name is replaced with **%s**.

Script Usage

Command format: **sh start-prequery.sh**

To run this command, place **user.keytab** or **jaas.conf** (either of them) and **krb5.conf** (mandatory) in the **conf** directory.

NOTE

- Currently, this tool supports only Carbon tables.
- This tool initializes the Carbon environment and pre-reads table metadata to JDBCServer. Therefore, this tool is more suitable for multi-active instances and static allocation mode.

24.16 Common Issues About Spark

24.16.1 Spark Core

24.16.1.1 How Do I View Aggregated Spark Application Logs?

Question

How do I view the aggregated container logs on the page when the log aggregation function is enabled on YARN?

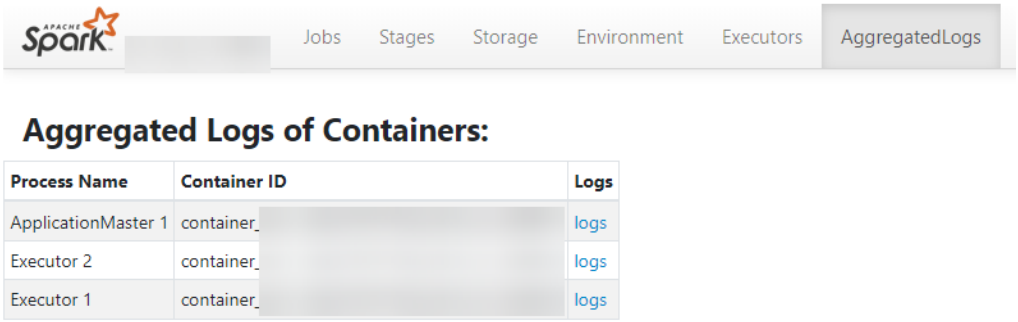
Answer

When **yarn.log-aggregation-enable** of YARN is set to **true**, the container log aggregation function is enabled. Log aggregation indicates that after applications are run on YARN, NodeManager aggregates all container logs of the node to HDFS and deletes local logs. For details, see **Configuring Container Log Aggregation**.

However, all logs will be aggregated to an HDFS directory and can only be viewed by accessing an HDFS file. Open-source Spark and YARN do not support the function of viewing aggregated logs on the web UI.

Spark supports this function. As shown in **Figure 24-10**, the **AggregatedLogs** tab is added to the HistoryServer page. You can click **logs** to view aggregated logs.

Figure 24-10 Aggregated log page



24.16.1.2 Why Is the Return Code of Driver Inconsistent with Application State Displayed on ResourceManager WebUI?

Question

Communication between ApplicationMaster and ResourceManager remains abnormal for a long time. Why is the driver return code inconsistent with application status on ResourceManager WebUI?

Answer

In yarn-client mode, Spark Driver and ApplicationMaster run as two independent processes. When Driver exits, it notifies ApplicationMaster to call the unregister API to deregister itself with ResourceManager.

This is a remote call and susceptible to network faults. If there exists a network fault, ApplicationMaster uses the retry mechanism of the YARN client to try again. If the network is recovered before the maximum number of retries is reached, ApplicationMaster exits gracefully.

If the number and duration of retries are reached, ApplicationMaster fails to deregister itself, and ResourceManager declares ApplicationMaster to have exited forcibly and tries to restart ApplicationMaster. After the restart, if ApplicationMaster fails to connect to the exited Driver, ResourceManager flags the Application being failed.

This problem rarely occurs and it does not impact the display of application states by SparkSQL. You can also increase the number of YARN client connections and the connection duration to reduce the probability of this event.

For details about the configuration, visit the following:

Versions earlier than MRS 3.2.0: **http://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-common/yarn-default.xml**

MRS 3.2.0 or later: **https://hadoop.apache.org/docs/r3.3.1/hadoop-yarn/hadoop-yarn-common/yarn-default.xml**

24.16.1.3 Why Cannot Exit the Driver Process?

Question

Why cannot exit the Driver process after running the **yarn application -kill applicationID** command to stop the Spark Streaming application?

Answer

Running the **yarn application -kill applicationID** command can only stop the SparkContext corresponding to Spark Streaming application, but cannot exit the current Driver process. If there are other permanent threads in the Driver process (for example, the spark shell is continually checking command input or Spark Streaming is continually reading data form data source), the Driver process will not be killed when the SparkContext is stopped. To exit the Driver process, you are advised to run the **kill -9 pid** command to kill the current Driver process by hand.

24.16.1.4 Why Does FetchFailedException Occur When the Network Connection Is Timed out

Question

On a large cluster of 380 nodes, run the ScalaSort test case in the HiBench test that runs the 29T data, and configure Executor as **--executor-cores 4**. The following abnormality is displayed:

```
org.apache.spark.shuffle.FetchFailedException: Failed to connect to /192.168.114.12:23242
    at
org.apache.spark.storage.ShuffleBlockFetcherIterator.throwFetchFailedException(ShuffleBlockFetcherIterator.s
cala:321)
    at org.apache.spark.storage.ShuffleBlockFetcherIterator.next(ShuffleBlockFetcherIterator.scala:306)
    at org.apache.spark.storage.ShuffleBlockFetcherIterator.next(ShuffleBlockFetcherIterator.scala:51)
    at scala.collection.Iterator$$anon$11.next(Iterator.scala:328)
    at scala.collection.Iterator$$anon$13.hasNext(Iterator.scala:371)
    at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:327)
    at org.apache.spark.util.CompletionIterator.hasNext(CompletionIterator.scala:32)
    at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:39)
    at org.apache.spark.util.collection.ExternalSorter.insertAll(ExternalSorter.scala:217)
    at org.apache.spark.shuffle.hash.HashShuffleReader.read(HashShuffleReader.scala:102)
    at org.apache.spark.rdd.ShuffledRDD.compute(ShuffledRDD.scala:90)
    at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:301)
    at org.apache.spark.rdd.RDD.iterator(RDD.scala:265)
    at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38)
    at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:301)
    at org.apache.spark.rdd.RDD.iterator(RDD.scala:265)
    at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38)
    at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:301)
    at org.apache.spark.rdd.RDD.iterator(RDD.scala:265)
    at org.apache.spark.rdd.UnionRDD.compute(UnionRDD.scala:87)
    at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:301)
    at org.apache.spark.rdd.RDD.iterator(RDD.scala:265)
    at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:73)
    at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:41)
    at org.apache.spark.scheduler.Task.run(Task.scala:87)
    at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:213)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
```



```
    at java.lang.Thread.run(Thread.java:745)
Caused by: java.io.IOException: Failed to connect to /192.168.114.12:23242
    at org.apache.spark.network.client.TransportClientFactory.createClient(TransportClientFactory.java:214)
    at org.apache.spark.network.client.TransportClientFactory.createClient(TransportClientFactory.java:167)
    at org.apache.spark.network.netty.NettyBlockTransferService$$anon
$1.createAndStart(NettyBlockTransferService.scala:91)
    at
org.apache.spark.network.shuffle.RetryingBlockFetcher.fetchAllOutstanding(RetryingBlockFetcher.java:140)
    at org.apache.spark.network.shuffle.RetryingBlockFetcher.access$200(RetryingBlockFetcher.java:43)
    at org.apache.spark.network.shuffle.RetryingBlockFetcher$1.run(RetryingBlockFetcher.java:170)
    at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    ... 3 more
Caused by: java.net.ConnectException: Connection timed out: /192.168.114.12:23242
    at sun.nio.ch.SocketChannelImpl.checkConnect(Native Method)
    at sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:717)
    at io.netty.channel.socket.nio.NioSocketChannel.doFinishConnect(NioSocketChannel.java:224)
    at io.netty.channel.nio.AbstractNioChannel
$AbstractNioUnsafe.finishConnect(AbstractNioChannel.java:289)
    at io.netty.channel.nio.NioEventLoop.processSelectedKey(NioEventLoop.java:528)
    at io.netty.channel.nio.NioEventLoop.processSelectedKeysOptimized(NioEventLoop.java:468)
    at io.netty.channel.nio.NioEventLoop.processSelectedKeys(NioEventLoop.java:382)
    at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:354)
    at io.netty.util.concurrent.SingleThreadEventExecutor$2.run(SingleThreadEventExecutor.java:111)
    ... 1 more
```

Answer

When an application is run, configure the Executor parameter as **--executor-cores 4**. The degree of parallelism (DOP) is high in a single process, resulting in that the IO is highly occupied and the task works slowly.

```
16/02/26 10:04:53 INFO TaskSetManager: Finished task 2139.0 in stage 1.0 (TID 151149) in 376455 ms on
10-196-115-2 (694/153378)
```

Because running a single task takes more than 6 minutes. The network connection is timed out and the running task fails.

Set the number of cores as 1, which is **--executor-cores 1**. A task is executed smoothly in proper time (within 15s).

```
16/02/29 02:24:46 INFO TaskSetManager: Finished task 59564.0 in stage 1.0 (TID 208574) in 15088 ms on
10-196-115-6 (59515/153378)
```

Therefore, to process the task of network connection timed out and avoid such error, you can reduce the core number of a single Executor.

24.16.1.5 How to Configure Event Queue Size If Event Queue Overflows?

Question

How to configure the event queue size if the following Driver log information is displayed indicating that the event queue overflows?

- Common applications

  ```
  Dropping SparkListenerEvent because no remaining room in event queue.
  This likely means one of the SparkListeners is too slow and cannot keep
  up with the rate at which tasks are being started by the scheduler.
  ```

- Spark Streaming applications

  ```
  Dropping StreamingListenerEvent because no remaining room in event queue.
  This likely means one of the StreamingListeners is too slow and cannot keep
  up with the rate at which events are being started by the scheduler.
  ```


Answer

1. Stop the application. Set the configuration option **spark.event.listener.logEnable** in the Spark configuration file **spark-defaults.conf** to **true**. And set the configuration option **spark.eventQueue.size** to **1000W**. If you need to control the logging rate (in milliseconds), also change the value of the configuration option **spark.event.listener.logRate**.

   By default, the logging rate is 1000 ms, which means that one log is printed out every 1000 ms.

2. Start the application.

   The following log information is displayed, including the event consumption rate, event production rate, and **MaxSize** (maximum size of messages in the queue).

   ```
   INFO LiveListenerBus: [SparkListenerBus]:16044 events are consumed in 5000 ms.
   INFO LiveListenerBus: [SparkListenerBus]:51381 events are produced in 5000 ms, eventQueue still has 86417 events, MaxSize: 171764.
   ```

3. Change the value of the configuration option **spark.eventQueue.size** in the Spark configuration file **spark-defaults.conf** based on the **MaxSize** in the log information.

   For example, if **MaxSize** is 250000, the appropriate message queue size is 300000.

24.16.1.6 What Can I Do If the getApplicationReport Exception Is Recorded in Logs During Spark Application Execution and the Application Does Not Exit for a Long Time?

Question

During Spark application execution, if the driver fails to connect to ResourceManager, the following error is reported and it does not exit for a long time. What can I do?

```
16/04/23 15:31:44 INFO RetryInvocationHandler: Exception while invoking getApplicationReport of class ApplicationClientProtocolPBClientImpl over 37 after 1 fail over attempts. Trying to fail over after sleeping for 44160ms.
java.net.ConnectException: Call From vm1/192.168.39.30 to vm1:8032 failed on connection exception: java.net.ConnectException: Connection refused; For more details see:  http://wiki.apache.org/hadoop/ConnectionRefused
```

Answer

In Spark, there is a scheduled thread that listens to the status of ApplicationMaster by connecting to ResourceManager. The connection to the ResourceManager times out. As a result, the preceding error is reported and the system keeps trying to connect to the ResourceManager. In the ResourceManager, the number of retry times is limited. By default, the number of retry times is 30 and the retry interval is about 30 seconds. The preceding error is reported during each retry. The driver exits only after the number of times is exceeded.

Table 24-77 describes the retry-related configuration items in the ResourceManager.

Table 24-77 Parameter description

Parameter	Description	Default Value
yarn.resourcemanager.connect.max-wait.ms	Maximum waiting time for connecting to the ResourceManager.	900000
yarn.resourcemanager.connect.retry-interval.ms	Interval for reconnecting to the ResourceManager.	30000

Number of retries (**yarn.resourcemanager.connect.max-wait.ms/ yarn.resourcemanager.connect.retry-interval.ms**) = Maximum waiting time for connecting to the ResourceManager/Interval for reconnecting to the ResourceManager

On the Spark client, modify the **conf/yarn-site.xml** file to add and configure **yarn.resourcemanager.connect.max-wait.ms** and **yarn.resourcemanager.connect.retry-interval.ms**. In this way, the number of retry times can be changed, and the Spark application can exit in advance.

24.16.1.7 What Can I Do If "Connection to ip:port has been quiet for xxx ms while there are outstanding requests" Is Reported When Spark Executes an Application and the Application Ends?

Question

When Spark executes an application, an error similar to the following is reported and the application ends. What can I do?

```
2016-04-20 10:42:00,557 | ERROR | [shuffle-server-2] | Connection to 10-91-8-208/10.18.0.115:57959 has been quiet for 180000 ms while there are outstanding requests. Assuming connection is dead; please adju st spark.network.timeout if this is wrong. | org.apache.spark.network.server.TransportChannelHandler.userEventTriggered(TransportChannelHandler.java:128)
2016-04-20 10:42:00,558 | ERROR | [shuffle-server-2] | Still have 1 requests outstanding when connection from 10-91-8-208/10.18.0.115:57959 is closed | org.apache.spark.network.client.TransportResponseHandler.channelUnregistered(TransportResponseHandler.java:102)
2016-04-20 10:42:00,562 | WARN  | [yarn-scheduler-ask-am-thread-pool-160] | Error sending message [message = DoShuffleClean(application_1459995017785_0108,319)] in 1 attempts | org.apache.spark.Logging$clas
s.logWarning(Logging.scala:92)
java.io.IOException: Connection from 10-91-8-208/10.18.0.115:57959 closed
        at org.apache.spark.network.client.TransportResponseHandler.channelUnregistered(TransportResponseHandler.java:104)
        at org.apache.spark.network.server.TransportChannelHandler.channelUnregistered(TransportChannelHandler.java:94)
        at io.netty.channel.AbstractChannelHandlerContext.invokeChannelUnregistered(AbstractChannelHandlerContext.java:158)
        at io.netty.channel.AbstractChannelHandlerContext.fireChannelUnregistered(AbstractChannelHandlerContext.java:144)
        at io.netty.channel.ChannelInboundHandlerAdapter.channelUnregistered(ChannelInboundHandlerAdapter.java:5
```



```
3)
        at
io.netty.channel.AbstractChannelHandlerContext.invokeChannelUnregistered(AbstractChannelHandlerContex
t.java:158)
        at
io.netty.channel.AbstractChannelHandlerContext.fireChannelUnregistered(AbstractChannelHandlerContext.ja
va:144)
        at
io.netty.channel.ChannelInboundHandlerAdapter.channelUnregistered(ChannelInboundHandlerAdapter.java:5
3)
        at
io.netty.channel.AbstractChannelHandlerContext.invokeChannelUnregistered(AbstractChannelHandlerContex
t.java:158)
        at
io.netty.channel.AbstractChannelHandlerContext.fireChannelUnregistered(AbstractChannelHandlerContext.ja
va:144)
        at
io.netty.channel.ChannelInboundHandlerAdapter.channelUnregistered(ChannelInboundHandlerAdapter.java:5
3)
        at
io.netty.channel.AbstractChannelHandlerContext.invokeChannelUnregistered(AbstractChannelHandlerContex
t.java:158)
        at
io.netty.channel.AbstractChannelHandlerContext.fireChannelUnregistered(AbstractChannelHandlerContext.ja
va:144)
        at io.netty.channel.DefaultChannelPipeline.fireChannelUnregistered(DefaultChannelPipeline.java:739)
        at io.netty.channel.AbstractChannel$AbstractUnsafe$8.run(AbstractChannel.java:659)
        at io.netty.util.concurrent.SingleThreadEventExecutor.runAllTasks(SingleThreadEventExecutor.java:357)
        at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:357)
        at io.netty.util.concurrent.SingleThreadEventExecutor$2.run(SingleThreadEventExecutor.java:111)
        at java.lang.Thread.run(Thread.java:745)
2016-04-20 10:42:00,573 | INFO  | [dispatcher-event-loop-14] | Starting task 177.0 in stage 1492.0 (TID
1996351, linux-254, PROCESS_LOCAL, 2106 bytes) | org.apache.spark.Logging$class.logInfo(Logging.scala:
59)
2016-04-20 10:42:00,574 | INFO  | [task-result-getter-0] | Finished task 85.0 in stage 1492.0 (TID 1996259)
in 191336 ms on linux-254 (106/3000) | org.apache.spark.Logging$class.logInfo(Logging.scala:59)
2016-04-20 10:42:00,811 | ERROR | [Yarn application state monitor] | Yarn application has already exited
with state FINISHED! | org.apache.spark.Logging$class.logError(Logging.scala:75)
```

Answer

Symptom: The value of **spark.rpc.io.connectionTimeout** is less than the value of **spark.rpc.askTimeout**. In full GC or network delay scenarios, when the channel reaches the expiration time and still receives no response, the channel is terminated. When detecting that the channel is terminated, the AM considers the driver as disconnected, and the entire application is stopped.

Solution:

Set the parameter in the **spark-defaults.conf** file on the Spark client by running the **set** command. During parameter configuration, ensure that the channel expiration time (**spark.rpc.io.connectionTimeout**) is greater than or equal to the RPC response timeout (**spark.rpc.askTimeout**).

Table 24-78 Parameter description

Parameter	Description	Default Value
spark.rpc.askTimeout	RPC response timeout. If this parameter is not set, the value of **spark.network.timeout** is used by default.	120s

24.16.1.8 Why Do Executors Fail to be Removed After the NodeManeger Is Shut Down?

Question

If the NodeManager is shut down with the Executor dynamic allocation enabled, the Executors on the node where the NodeManeger is shut down fail to be removed from the driver page after the idle time expires.

Answer

When the ResourceManager detects that the NodeManager is shut down, the driver has requested to kill Executors due to idle time expiry. However, the Executors cannot actually be killed because the NodeManager is shut down.

The driver cannot detect the LOST events of these Executors and does not remove Executors from its Executor list. Therefore, the Executors are not removed from the driver page.

This phenomenon is normal after the YARN NodeManager is shut down. The Executors will be removed after the NodeManager restarts.

24.16.1.9 What Can I Do If the Message "Password cannot be null if SASL is enabled" Is Displayed?

Question

ExternalShuffle is enabled for the application that runs Spark. Task loss occurs in the application because the message "java.lang.NullPointerException: Password cannot be null if SASL is enabled" is displayed. The following shows some key logs:

```
2016-05-13 12:05:27,093 | WARN  | [task-result-getter-2] | Lost task 98.0 in stage 22.1 (TID 193603, linux-173, 2): FetchFailed(BlockManagerId(13, 172.168.100.13, 27337),
org.apache.spark.shuffle.FetchFailedException: java.lang.NullPointerException: Password cannot be null if SASL is enabled
        at org.spark-project.guava.base.Preconditions.checkNotNull(Preconditions.java:208)
        at org.apache.spark.network.sasl.SparkSaslServer.encodePassword(SparkSaslServer.java:196)
        at org.apache.spark.network.sasl.SparkSaslServer$DigestCallbackHandler.handle(SparkSaslServer.java:166)
        at com.sun.security.sasl.digest.DigestMD5Server.validateClientResponse(DigestMD5Server.java:589)
        at com.sun.security.sasl.digest.DigestMD5Server.evaluateResponse(DigestMD5Server.java:244)
        at org.apache.spark.network.sasl.SparkSaslServer.response(SparkSaslServer.java:119)
        at org.apache.spark.network.sasl.SaslRpcHandler.receive(SaslRpcHandler.java:100)
        at org.apache.spark.network.server.TransportRequestHandler.processRpcRequest(TransportRequestHandler.java:128)
        at org.apache.spark.network.server.TransportRequestHandler.handle(TransportRequestHandler.java:99)
        at org.apache.spark.network.server.TransportChannelHandler.channelRead0(TransportChannelHandler.java:104)
```

Answer

The cause is that NodeManager restarts. When ExternalShuffle is used, Spark uses NodeManager to transmit shuffle data. Therefore, the memory of NodeManager may be seriously insufficient.

In the FusionInsight of the current version, the default memory of NodeManager is only 1 GB. When the data volume of Spark tasks is large (greater than 1 TB), the memory is severely insufficient and the message response is slow. As a result, the FusionInsight health check determines that the NodeManager process exits and forcibly restarts the NodeManager, causing the preceding problem.

Solution:

Adjust the memory of the NodeManager. If the data volume is large (greater than 1 TB), the memory of NodeManager must be greater than 4 GB.

24.16.1.10 "Failed to CREATE_FILE" Is Displayed When Data Is Inserted into the Dynamic Partitioned Table Again

Question

When inserting data into a dynamically partitioned table, shuffle file corruption (due to issues like disk disconnections or node failures) can lead to a "Failed to CREATE_FILE" exception during task retries.

```
2016-06-25 15:11:31,323 | ERROR | [Executor task launch worker-0] | Exception in task 15.0 in stage 10.1 (TID 1258) | org.apache.spark.Logging$class.logError(Logging.scala:96)
org.apache.hadoop.hive.ql.metadata.HiveException:
org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.hdfs.protocol.AlreadyBeingCreatedException):
Failed to CREATE_FILE /user/hive/warehouse/testdb.db/we
b_sales/.hive-staging_hive_2016-06-25_15-09-16_999_8137121701603617850-1/-ext-10000/_temporary/0/
_temporary/attempt_201606251509_0010_m_000015_0/ws_sold_date=1999-12-17/part-00015 for
DFSClient_attempt_2016
06251509_0010_m_000015_0_353134803_151 on 10.1.1.5 because this file lease is currently owned by
DFSClient_attempt_201606251509_0010_m_000015_0_-848353830_156 on 10.1.1.6
```

Answer

The last step of inserting data into a dynamically partitioned table is to read data from the shuffle file and write the data to the partition file corresponding to the table.

If a large number of shuffle files are damaged, a large number of tasks fail and jobs are retried. Before the retry, Spark closes the handle for writing table partition files. If a significant number of tasks close their handles, HDFS may not be able to process them promptly. When the task is retried next time, the handle is not released on the NameNode in a timely manner. As a result, the "Failed to CREATE_FILE" exception occurs.

However, this issue is typically transient and has minimal impact, as retries occur within milliseconds.

24.16.1.11 Why Tasks Fail When Hash Shuffle Is Used?

Question

When Hash shuffle is used to run a job that consists of 1000000 map tasks x 100000 reduce tasks, run logs report many message failures and Executor heartbeat timeout, leading to task failures. Why does this happen?

Answer

During the shuffle process, Hash shuffle just writes the data of different reduce partitions to their respective disk files according to hash results without sorting the data.

If there are many reduce partitions, a large number of disk files will be generated. In your case, 10^11 shuffle files, that is, 1000000 * 100000 shuffle files, will be generated. The sheer number of disk files will have a great impact on the file read and write performance. In addition, the operations such as sorting and compressing will consume a large amount of temporary memory space because a large number of file handles are open, presenting great challenges to memory

management and garbage collection and incurring the possibility that the Executor fails to respond to Driver.

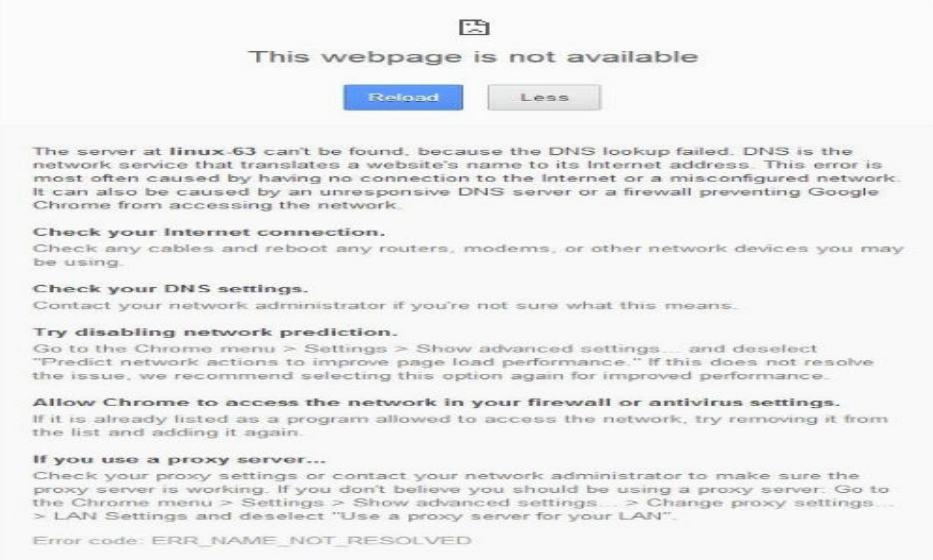
Sort shuffle, instead of Hash shuffle, is recommended to run a job.

24.16.1.12 What Can I Do If the Error Message "DNS query failed" Is Displayed When I Access the Aggregated Logs Page of Spark Applications?

Question

When the **http(s)://<spark ip>:<spark port>** mode is used to access the Spark JobHistory page, if the displayed Spark JobHistory page is not the page of FusionInsight Manager (the URL of FusionInsight Manager is similar to **https://<oms ip>:20026/Spark2x/JobHistory2x/xx/**), click an application and click **AggregatedLogs**, click the logs of an executor to be viewed. An error message in **Figure 24-11** is displayed.

Figure 24-11 DNS query failure



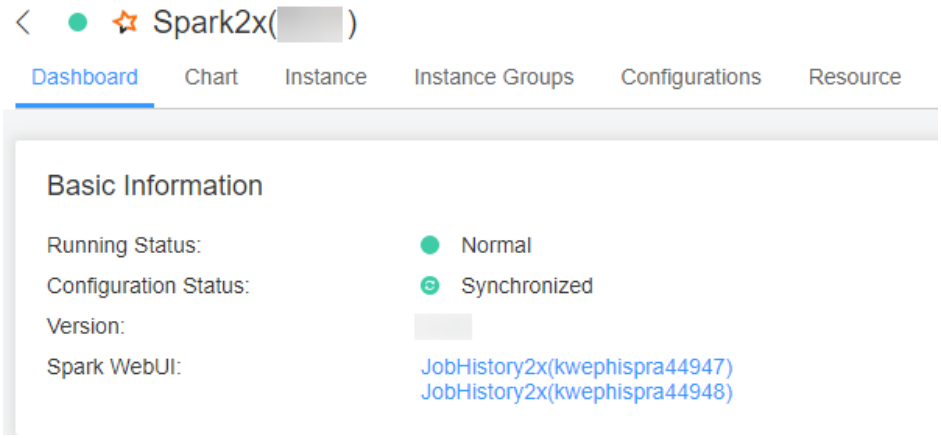
Answer

Cause: The domain name is not added to the **hosts** file of the Windows OS in the pop-up URL (for example, **https://<hostname>:20026/Spark2x/JobHistory2x/xx/history/application_xxx/jobs/**). As a result, the DNS query fails and the web page cannot be displayed.

Solution:

- You are advised to visit the **Spark JobHistory** page using FusionInsight Manager. Click the links in the blue box in **Figure 24-12**.

Figure 24-12 Spark2x page of FusionInsight Manager



- If you do not want to access the **Spark JobHistory** page using the FusionInsight Manager, change **<hostname>** in the URL to the IP address or add the domain name to the **hosts** file of the Windows OS.

24.16.1.13 What Can I Do If Shuffle Fetch Fails Due to the "Timeout Waiting for Task" Exception?

Question

When I execute a 100 TB TPC-DS test suite in the JDBCServer mode, the "Timeout waiting for task" is displayed. As a result, shuffle fetch fails, the stage keeps retrying, and the task cannot be completed properly. What can I do?

Answer

The ShuffleService function is used in JDBCServer mode. In the reduce phase, all executors obtain data from NodeManager. When the data volume reaches a level (more than 10 TB), the NodeManager may reach the bottleneck (ShuffleService is in the NodeManager process). As a result, some tasks for obtaining data time out. Therefore, the problem occurs.

You are advised to disable ShuffleService for Spark tasks whose data volume is greater than 10 TB. That is, set **spark.shuffle.service.enabled** in the **Spark-defaults.conf** configuration file to **false**.

24.16.1.14 Why Does the Stage Retry due to the Crash of the Executor?

Question

When I run Spark tasks with a large data volume, for example, 100 TB TPCDS test suite, why does the Stage retry due to Executor loss sometimes? The message "Executor 532 is lost rpc with driver, but is still alive, going to kill it" is displayed, indicating that the loss of the Executor is caused by a JVM crash.

The log of the key JVM crash is as follows:

```
#
# A fatal error has been detected by the Java Runtime Environment:
#
```



```
#  Internal Error (sharedRuntime.cpp:834), pid=241075, tid=140476258551552
#  fatal error: exception happened outside interpreter, nmethods and vtable stubs at pc
0x00007fcda9eb8eb1
```

Answer

This error does not affect services. This error is caused by defects of the Oracle JVM, but not the platform code. There is the fault tolerance mechanism for Executors in Spark: the Stage retries in case of an Executor crash to ensure the success execution of tasks.

24.16.1.15 Why Do the Executors Fail to Register Shuffle Services During the Shuffle of a Large Amount of Data?

Question

When more than 50 terabytes of data is shuffled, some executors fail to register shuffle services due to timeout. The shuffle tasks then fail. Why? The error log is as follows:

```
2016-10-19 01:33:34,030 | WARN | ContainersLauncher #14 | Exception from container-launch with
container ID: container_e1452_1476801295027_2003_01_004512 and exit code: 1 |
LinuxContainerExecutor.java:397
ExitCodeException exitCode=1:
at org.apache.hadoop.util.Shell.runCommand(Shell.java:561)
at org.apache.hadoop.util.Shell.run(Shell.java:472)
at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.java:738)
at
org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor.launchContainer(LinuxContainerExecuto
r.java:381)
at
org.apache.hadoop.yarn.server.nodemanager.containermanager.launcher.ContainerLaunch.call(ContainerLaun
ch.java:312)
at
org.apache.hadoop.yarn.server.nodemanager.containermanager.launcher.ContainerLaunch.call(ContainerLaun
ch.java:88)
at java.util.concurrent.FutureTask.run(FutureTask.java:266)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
2016-10-19 01:33:34,031 | INFO | ContainersLauncher #14 | Exception from container-launch. |
ContainerExecutor.java:300
2016-10-19 01:33:34,031 | INFO | ContainersLauncher #14 | Container id:
container_e1452_1476801295027_2003_01_004512 | ContainerExecutor.java:300
2016-10-19 01:33:34,031 | INFO | ContainersLauncher #14 | Exit code: 1 | ContainerExecutor.java:300
2016-10-19 01:33:34,031 | INFO | ContainersLauncher #14 | Stack trace: ExitCodeException exitCode=1: |
ContainerExecutor.java:300
```

Answer

The imported data exceeds 50 TB, which exceeds the shuffle processing capability. The shuffle may fail to respond to the registration request of an executor in a timely manner due to the heavy load.

The timeout interval for an executor to register the shuffle service is 5 seconds. The maximum number of retries is 3. This parameter is not configurable.

You are advised to increase the number of task retry times and the number of allowed executor failure times.

Configure the following parameters in the **spark-defaults.conf** file on the client: If **spark.yarn.max.executor.failures** does not exist, manually add it.

Table 24-79 Parameter Description

Parameter	Description	Default Value
spark.task.maxFailures	Specifies task retry times.	4
spark.yarn.max.executor.failures	Specifies executor failure attempt times. Set **spark.dynamicAllocation.enabled** to **false**, to disable the dynamic allocation of executors.	numExecutors * 2, with minimum of 3
	Specifies executor failure attempt times. Set **spark.dynamicAllocation.enabled** to **true**, to enable the dynamic allocation of executors.	3

24.16.1.16 NodeManager OOM Occurs During Spark Application Execution

Question

Enabling YARN's External Shuffle Service and having an excessive number of shuffle connections during the execution of a Spark application will trigger error "java.lang.OutofMemoryError: Direct buffer Memory". This indicates that the memory is insufficient. The error log is as follows:

```
2016-12-06 02:01:00,768 | WARN  | shuffle-server-38 | Exception in connection from /192.168.101.95:53680 | TransportChannelHandler.java:79
io.netty.handler.codec.DecoderException: java.lang.OutOfMemoryError: Direct buffer memory
        at io.netty.handler.codec.ByteToMessageDecoder.channelRead(ByteToMessageDecoder.java:153)
        at io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.java:333)
        at io.netty.channel.AbstractChannelHandlerContext.fireChannelRead(AbstractChannelHandlerContext.java:319)
        at io.netty.channel.DefaultChannelPipeline.fireChannelRead(DefaultChannelPipeline.java:787)
        at io.netty.channel.nio.AbstractNioByteChannel$NioByteUnsafe.read(AbstractNioByteChannel.java:130)
        at io.netty.channel.nio.NioEventLoop.processSelectedKey(NioEventLoop.java:511)
        at io.netty.channel.nio.NioEventLoop.processSelectedKeysOptimized(NioEventLoop.java:468)
        at io.netty.channel.nio.NioEventLoop.processSelectedKeys(NioEventLoop.java:382)
        at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:354)
        at io.netty.util.concurrent.SingleThreadEventExecutor$2.run(SingleThreadEventExecutor.java:116)
        at java.lang.Thread.run(Thread.java:745)
Caused by: java.lang.OutOfMemoryError: Direct buffer memory
        at java.nio.Bits.reserveMemory(Bits.java:693)
        at java.nio.DirectByteBuffer.<init>(DirectByteBuffer.java:123)
        at java.nio.ByteBuffer.allocateDirect(ByteBuffer.java:311)
        at io.netty.buffer.PoolArena$DirectArena.newChunk(PoolArena.java:434)
        at io.netty.buffer.PoolArena.allocateNormal(PoolArena.java:179)
        at io.netty.buffer.PoolArena.allocate(PoolArena.java:168)
        at io.netty.buffer.PoolArena.reallocate(PoolArena.java:277)
        at io.netty.buffer.PooledByteBuf.capacity(PooledByteBuf.java:108)
```



```
    at io.netty.buffer.AbstractByteBuf.ensureWritable(AbstractByteBuf.java:251)
    at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:849)
    at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:841)
    at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:831)
    at io.netty.handler.codec.ByteToMessageDecoder.channelRead(ByteToMessageDecoder.java:146)
    ... 10 more
```

Answer

YARN's External Shuffle Service starts twice the number of threads as there are available vCPUs. However, the default direct buffer memory is only 128 MB. This means that when a large number of shuffle connections are established simultaneously, the direct buffer memory allocated to each thread is low. For instance, if a node has 40 vCPUs, YARN's External Shuffle Service will start 80 threads, and these threads will share the direct buffer memory. As a result, the memory allocated to each thread will be less than 2 MB.

So, you are advised to adjust the value of direct buffer memory based on the number of vCPUs of the NodeManager node in the cluster. For example, if the number of vCPUs is 40, set direct buffer memory to 512 MB. That is, set the **GC_OPTS** parameter of the NodeManager node. The following is an example:

-XX:MaxDirectMemorySize=512M

NOTE

The **-XX:MaxDirectMemorySize** parameter is not used by default. If you need to set this parameter, add it to the **GC_OPTS** parameter.

Perform the following operations to configure this parameter:

Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Yarn**. Click **Configurations**, click **All Configurations**, click **NodeManager**, and select **System**. Then, modify the configuration in the **GC_OPTS** parameter in the right pane.

Table 24-80 Parameters

Parameter	Description	Default Value
GC_OPTS	GC parameter of YARN NodeManager	128M

24.16.2 Spark SQL and DataFrame

24.16.2.1 What Do I have to Note When Using Spark SQL ROLLUP and CUBE?

Question

Suppose that there is a table src(d1, d2, m) with the following data:

```
1 a 1
1 b 1
2 b 2
```


The results for statement "select d1, sum(d1) from src group by d1, d2 with rollup" are shown as below:

```
NULL 0
1    2
2    2
1    1
1    1
2    2
```

Why the first line of the above results is (NULL,0), rather than (NULL,4)?

Answer

When conducting the rollup and cube operation, we usually perform the dimension-based analysis and what we need is the measurement result, so we would not conduct aggregation operation on the dimension.

Suppose that there is a table src(d1, d2, m), so the statement 1 "select d1, sum(m) from src group by d1, d2 with rollup" conducts the rollup operation on the dimension d1 and d2 to compute the result of m. It has actual business meaning, and its results are in line with the expectation. However, the statement 2 "select d1, sum(d1) from src group by d1, d2 with rollup" cannot be explained from the business perspective. For the statement 2, the result for all aggregations (sum/avg/max/min) is 0.

NOTE

Only when there is an aggregation operation for fields in "group by" in the rollup and cube operation, the result is 0. For non-rollup and non-cube operations, the result will be in line with the expectation.

24.16.2.2 Why Spark SQL Is Displayed as a Temporary Table in Different Databases?

Question

Why temporary tables of the previous database are displayed after the database is switched?

1. Create a temporary DataSource table, for example:

   ```
   create temporary table ds_parquet
   using org.apache.spark.sql.parquet
   options(path '/tmp/users.parquet');
   ```

2. Switch to another database, and run ***show tables***. The temporary table created in the previous table is displayed.

   ```
   0: jdbc:hive2://192.168.169.84:22550/default> show tables;
   +-----------------+--------------+--+
   |    tableName    | isTemporary  |
   +-----------------+--------------+--+
   | ds_parquet      | true         |
   | cmb_tbl_carbon  | false        |
   +-----------------+--------------+--+
   2 rows selected (0.109 seconds)
   0: jdbc:hive2://192.168.169.84:22550/default>
   ```

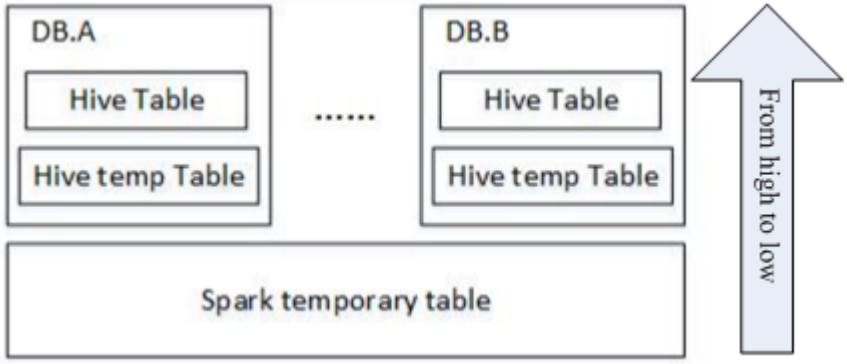

Answer

The table management hierarchy of Spark is shown in **Figure 24-13**. The lowest layer stores all temporary DataSource tables. There is no such concept as database at this layer. DataSource tables are visible in various databases.

The MetaStore of Hive is located at the upper layer. This layer distinguishes among databases. In each database, there are two types of Hive table, permanent and temporary. Therefore, Spark supports data tables of the same name at three layers.

During query, SparkSQL first checks for temporary Spark tables, then temporary Hive tables in the current database, and at last the permanent tables in the current database.

Figure 24-13 Spark table management hierarchy



When a session quits, temporary tables related to the user operation are automatically deleted. Manual deletion of temporary files is not recommended.

When deleting temporary files, use the same priority as that for query. The priorities are temporary Spark table, temporary Hive table, and permanent Hive table ranging from high to low. If you want to directly delete Hive tables but not temporary Spark tables, you can directly use the ***drop table DbName.TableName*** command.

24.16.2.3 How to Assign a Parameter Value in a Spark Command?

Question

I do not like the idea to assign parameter values on user interface or in configuration files. How to do it with Spark commands?

Answer

Spark configuration options can be defined either in a configuration file or in Spark commands.

To assign a parameter value, run the **--conf** *Parameter name*=*Parameter value* command on a Spark client. Add the parameter name and its value after **--conf**. The following is an example.

```
--conf spark.eventQueue.size=50000
```


24.16.2.4 What Directory Permissions Do I Need to Create a Table Using SparkSQL?

Question

The following error information is displayed when a new user creates a table using SparkSQL:

```
0: jdbc:hive2://192.168.169.84:22550/default> create table testACL(c string);
Error: org.apache.spark.sql.execution.QueryExecutionException: FAILED: Execution Error, return code 1 from
org.apache.hadoop.hive.ql.exec.DDLTask. MetaException(message:Got exception:
org.apache.hadoop.security.AccessControlException
Permission denied: user=testACL, access=EXECUTE, inode="/user/hive/warehouse/
testacl":spark:hadoop:drwxrwx---
    at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkAccessAcl(FSPermissionChecker.java:403
)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:306)
    at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkTraverse(FSPermissionChecker.java:259)
    at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:20
5)
    at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:19
0)
    at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermission(FSDirectory.java:1710)
    at
org.apache.hadoop.hdfs.server.namenode.FSDirStatAndListingOp.getFileInfo(FSDirStatAndListingOp.java:109)
    at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getFileInfo(FSNamesystem.java:3762)
    at
org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.getFileInfo(NameNodeRpcServer.java:1014)
    at
org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTranslatorPB.getFileInfo(ClientNamen
odeProtocolServerSideTranslatorPB.java:853)
    at org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$ClientNamenodeProtocol
$2.callBlockingMethod(ClientNamenodeProtocolProtos.java)
    at org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.call(ProtobufRpcEngine.java:616)
    at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:973)
    at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2089)
    at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2085)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:422)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1675)
    at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2083)
) (state=,code=0)
```

Answer

When you create a table using Spark SQL, the interface of Hive is called by the underlying system and a directory named after the table will be created in the **/user/hive/warehouse** directory. Therefore, you must have the permissions to read, write, and execute the **/user/hive/warehouse** directory or the group permission of Hive.

The**/user/hive/warehouse** is specified by the hive.metastore.warehouse.dir parameter.

24.16.2.5 Why Do I Fail to Delete the UDF Using Another Service?

Question

Why do I fail to delete the UDF using another service, for example, delete the UDF created by Hive using Spark SQL.

Answer

The UDF can be created using any of the following services:

1. Hive client.
2. JDBCServer API. You can connect JDBCServer to Spark Beeline or JDBC client code, and run SQL statements to create the UDF.
3. spark-sql.

The scenarios in which the UDF failed to be deleted may be as follows:

- If you use Spark Beeline to delete the UDF created by other services, you must restart the JDBCServer before the deletion. Otherwise, the deletion fails. If you use spark-sql to delete the UDF created by other services, you must restart the spark-sql before the deletion. Otherwise, the deletion fails.

  Cause: After the UDF is created, if the JDBCServer or the spark-sql has not been restarted, the newly created UDF will not be saved by the FunctionRegistry object in the thread where Spark locates. As a result, the UDF failed to be deleted.

  Solution: Restart the JDBCServer and spark-sql of the Spark client and delete the UDF.

- When creating UDF on the Hive client, the **add jar** command (e.g. **add jar /opt/test/two_udfs.jar**) is used to add the **.jar** package instead of specifying the path of **.jar** package in creating UDF statement. As a result, the **ClassNotfound** error occurs when you use other services to delete the UDF.

  Cause: When you use a service to delete the UDF, the service will load the class that corresponds to the UDF to obtain the UDF. However, the **.jar** package is added by the ***add jar*** command and jar package does not exist in the classpath of other services. As a result, the **ClassNotfound** error occurs and the UDF failed to be deleted.

  Solution: The UDF created using the preceding approach must be deleted using the same approach. No other approaches are allowed.

24.16.2.6 Why Cannot I Query Newly Inserted Data in a Parquet Hive Table Using SparkSQL?

Question

Why cannot I query newly inserted data in a parquet Hive table using SparkSQL? This problem occurs in the following scenarios:

1. For partitioned tables and non-partitioned tables, after data is inserted on the Hive client, the latest inserted data cannot be queried using SparkSQL.

2. After data is inserted into a partitioned table using SparkSQL, if the partition information remains unchanged, the newly inserted data cannot be queried using SparkSQL.

Answer

To improve Spark performance, parquet metadata is cached. When the parquet table is updated by Hive or another means, the cached metadata remains unchanged, resulting in SparkSQL failing to query the newly inserted data.

For a parquet Hive partition table, if the partition information remains unchanged after data is inserted, the cached metadata is not updated. As a result, the newly inserted data cannot be queried by SparkSQL.

To solve the query problem, update metadata before starting a Spark SQL query.

REFRESH TABLE table_name;

table_name indicates the name of the table to be updated. The table must exist. Otherwise, an error is reported.

When the query statement is executed, the latest inserted data can be obtained.

24.16.2.7 How to Use Cache Table?

Question

What is cache table used for? Which point should I pay attention to while using cache table?

Answer

Spark SQL caches tables into memory so that data can be directly read from memory instead of disks, reducing memory overhead due to disk reads.

Note that cached tables consume Executor's memory. This means that caching large or many tables compromises Executor's stability even if compressed storage has been used to reduce memory overhead as much as possible.

If it is no longer necessary to accelerate data query by means of cache table, run the following command to uncache tables to free up memory:

uncache table *table_name*

NOTE

The Storage tab page of the Spark Driver user interface displays the cached tables.

24.16.2.8 Why Are Some Partitions Empty During Repartition?

Question

During the repartition operation, the number of blocks (**spark.sql.shuffle.partitions**) is set to 4,500, and the number of keys used by repartition exceeds 4,000. It is expected that data corresponding to different keys

can be allocated to different partitions. However, only 2,000 partitions have data, and data corresponding to different keys is allocated to the same partition.

Answer

This is normal.

The partition to which data is distributed is obtained by performing a modulo operation on hashcode of a key. Different hashcodes may have the same modulo result. In this case, data is distributed to the same partition, as a result, some partitions do not have data, and some partitions have data corresponding to multiple keys.

You can adjust the value of **spark.sql.shuffle.partitions** to adjust the cardinality during modulo operation and improve the unevenness of data blocks. After multiple verifications, it is found that the effect is good when the parameter is set to a prime number or an odd number.

Configure the following parameters in the **spark-defaults.conf** file on the Driver client.

Table 24-81 Parameter Description

Parameter	Description	Default Value
spark.sql.shuffle.partitions	Number of shuffle data blocks during the shuffle operation.	200

24.16.2.9 Why Does 16 Terabytes of Text Data Fails to Be Converted into 4 Terabytes of Parquet Data?

Question

When the default configuration is used, 16 terabytes of text data fails to be converted into 4 terabytes of parquet data, and the error information below is displayed. Why?

```
Job aborted due to stage failure: Task 2866 in stage 11.0 failed 4 times, most recent failure: Lost task
2866.6 in stage 11.0 (TID 54863, linux-161, 2): java.io.IOException: Failed to connect to /10.16.1.11:23124
at org.apache.spark.network.client.TransportClientFactory.createClient(TransportClientFactory.java:214)
at org.apache.spark.network.client.TransportClientFactory.createClient(TransportClientFactory.java:167)
at org.apache.spark.network.netty.NettyBlockTransferService$$anon
$1.createAndStart(NettyBlockTransferService.scala:92)
```

Table 24-82 lists the default configuration.

Table 24-82 Parameter Description

Parameter	Description	Default Value
spark.sql.shuffle.partitions	Number of shuffle data blocks during the shuffle operation.	200

Parameter	Description	Default Value
spark.shuffle.sasl.timeout	Timeout interval of SASL authentication for the shuffle operation. Unit: second	120s
spark.shuffle.io.connectionTimeout	Timeout interval for connecting to a remote node during the shuffle operation. Unit: second	120s
spark.network.timeout	Timeout interval for all network connection operations. Unit: second	360s

Answer

The current data volume is 16 TB, but the number of partitions is only 200. As a result, each task is overloaded and the preceding problem occurs.

To solve the preceding problem, you need to adjust the parameters.

- Increase the number of partitions to divide the task into smaller ones.
- Increase the timeout interval during task execution.

Configure the following parameters in the **spark-defaults.conf** file on the client:

Table 24-83 Parameter Description

Parameter	Description	Recommended Value
spark.sql.shuffle.partitions	Number of shuffle data blocks during the shuffle operation.	4501
spark.shuffle.sasl.timeout	Timeout interval of SASL authentication for the shuffle operation. Unit: second	2000s
spark.shuffle.io.connectionTimeout	Timeout interval for connecting to a remote node during the shuffle operation. Unit: second	3000s
spark.network.timeout	Timeout interval for all network connection operations. Unit: second	360s

24.16.2.10 How Do I Rectify the Exception Occurred When I Perform an Operation on the Table Named table?

Question

After a table named **table** is created, the following error message is displayed when you run **drop table table** or perform other operations.

```
16/07/12 18:56:29 ERROR SparkSQLDriver: Failed in [drop table table]
java.lang.RuntimeException: [1.1] failure: identifier expected
table
^
at scala.sys.package$.error(package.scala:27)
at org.apache.spark.sql.catalyst.SqlParserTrait$class.parseTableIdentifier(SqlParser.scala:56)
at org.apache.spark.sql.catalyst.SqlParser$.parseTableIdentifier(SqlParser.scala:485)
```

Answer

table is a keyword of Spark SQL and cannot be used as a table name.

It is recommended that you do not use table as a **table** name when creating a table.

24.16.2.11 Why Is a Task Suspended When the ANALYZE TABLE Statement Is Executed and Resources Are Insufficient?

Question

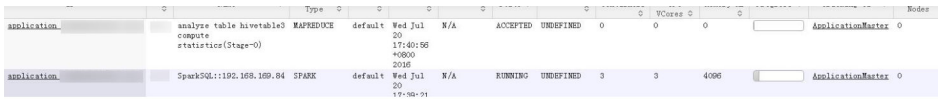
When the ***analyze table*** statement is executed using spark-sql, the task is suspended and the information below is displayed. Why?

```
spark-sql> analyze table hivetable2 compute statistics;
Query ID = root_20160716174218_90f55869-000a-40b4-a908-533f63866fed
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
16/07/20 17:40:56 WARN JobResourceUploader: Hadoop command-line option parsing not performed.
Implement the Tool interface and execute your application with ToolRunner to remedy this.
Starting Job = job_1468982600676_0002, Tracking URL = http://10-120-175-107:8088/proxy/
application_1468982600676_0002/
Kill Command = /opt/client/HDFS/hadoop/bin/hadoop job  -kill job_1468982600676_0002
```

Answer

When the statement is executed, the SQL statement starts the ***analyze table hivetable2 compute statistics*** MapReduce tasks. On the ResourceManager Web UI of Yarn, the task is not executed due to insufficient resources. As a result, the task is suspended.

Figure 24-14 ResourceManager web UI



You are advised to add **noscan** when running the ***analyze table*** statement. The function of this statement is the same as that of the ***analyze table hivetable2 compute statistics*** statement. The command is as follows:

```
spark-sql> analyze table hivetable2 compute statistics noscan
```

This command does not start MapReduce tasks and does not occupy Yarn resources. Therefore, the tasks can be executed.

24.16.2.12 If I Access a parquet Table on Which I Do not Have Permission, Why a Job Is Run Before "Missing Privileges" Is Displayed?

Question

If I access a parquet table on which I do not have permission, why a job is run before "Missing Privileges" is displayed?

Answer

The execution sequence of Spark SQL statement parse the table in the statement first, then obtain the metadata in the table, and finally check the permission.

The metadata of a parquet table contains the Split information (which is read by HDFS API) about files. If the table contains many files, the HDFS API reads data in serial mode, in which degrades the performance. If the number of files in the table exceeds the threshold *spark.sql.sources.parallelSplitDiscovery.threshold*, a job will be generated to use Executor to read the data in parallel mode.

The permission authentication is executed after the metadata is obtained. Therefore, when the number of files in the table exceeds the threshold, a job is run before the permission authentication error message **Missing Privileges**.

24.16.2.13 Why Is "RejectedExecutionException" Displayed When I Exit Spark SQL?

Question

After successfully running Spark tasks with large data volume, for example, 2-TB TPCDS test suite, why is the abnormal stack information **"RejectedExecutionException"** displayed sometimes? The log is as follows:

```
16/07/16 10:19:56 ERROR TransportResponseHandler: Still have 2 requests outstanding when connection from linux-192/10.1.1.5:59250 is closed
java.util.concurrent.RejectedExecutionException: Task scala.concurrent.impl.CallbackRunnable@5fc1ab rejected from java.util.concurrent.ThreadPoolExecutor@52fa7e19[Terminated, pool size = 0, active threads = 0, queued tasks = 0, completed tasks = 3025]
```

Answer

When Spark SQL is closed, the application and the message channel are closed. If there are unprocessed messages, the connection should be closed to rectify the exception. If the thread pool inside Scala is closed, the abnormal stack information "RejectedExecutionException" is displayed. This abnormal stack information will not be displayed if the thread pool inside Scala is not closed.

The error occurs when the application is successfully run and closed. Therefore, the error will not affect the services.

24.16.2.14 How Do I Do If I Incidentally Kill the JDBCServer Process During Health Check?

Question

In the health check solution, when the number of concurrently executed statements reaches the upper limit of the thread pool, the health check command fails to run. As a result, the health check program times out and the Spark JDBCServer process is killed.

Answer

Currently, JDBCServer has two thread pools **HiveServer2-Handler-Pool** and **HiveServer2-Background-Pool**. HiveServer2-Handler-Pool is used to process session connections, and HiveServer2-Background-Pool is used to execute SQL statements.

In the current health check mechanism, a session connection is created and the health check command **HEALTHCHECK** is executed in the thread where the session is located to determine the health status of Spark JDBCServer. Therefore, HiveServer2-Handler-Pool must reserve a thread to process the health check session connection and execute the health check command, otherwise, the health check session cannot be established or the health check command cannot be executed. As a result, Spark JDBCServer is regarded as unhealthy and then killed. That is, if the number of thread pools of HiveServer2-Handler-Pool is 100, a maximum of 99 sessions can be connected.

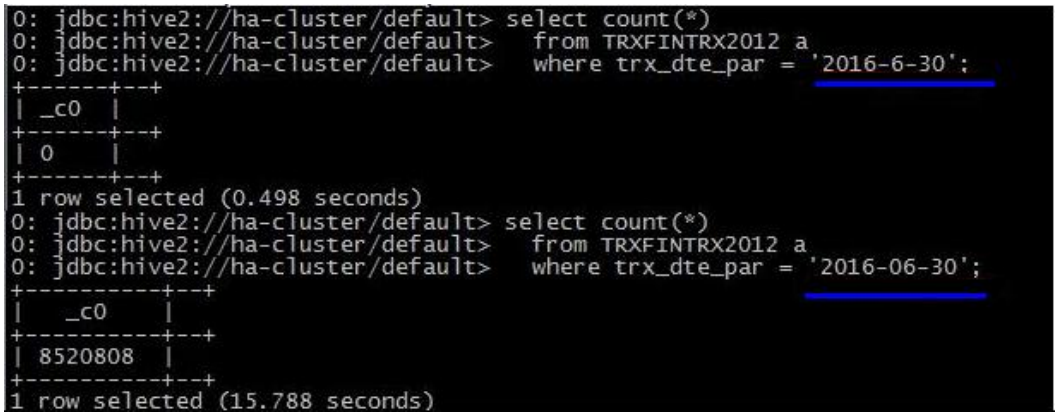
24.16.2.15 Why No Result Is found When 2016-6-30 Is Set in the Date Field as the Filter Condition?

Question

Why no result is found when 2016-6-30 is set in the date field as the filter condition?

As shown in the following figure, trx_dte_par in the select count (*) from trxfintrx2012 a where trx_dte_par='2016-6-30' statement is a date field. However, no search result is found when the filter condition is where trx_dte_par='2016-6-30'. Search results are found only when the filter condition is where trx_dte_par='2016-06-30'.

Figure 24-15 Example



Answer

If a data string of the date type is present in Spark SQL statements, the Spark SQL will search the matching character string without checking the date format. In this case, if the date format in the SQL statement is incorrect, the query will fail. For example, if the data format is yyyy-mm-dd, then no search results matching '2016-6-30' will be found.

24.16.2.16 Why Is the "Code of method ... grows beyond 64 KB" Error Message Displayed When I Run Complex SQL Statements?

Question

When I run a complex SQL statement, for example, SQL statements with multiple layers of nesting statements and a single layer statement contains a large number of logic clauses such as case when, an error message indicating that the code of a certain method exceeds 64 KB is displayed. The log is as follows:

```
java.util.concurrent.ExecutionException: java.lang.Exception: failed to compile:
org.codehaus.janino.JaninoRuntimeException: Code of method "(Lorg/apache/spark/sql/catalyst/expressions/
GeneratedClass$SpecificUnsafeProjection;Lorg/apache/spark/sql/catalyst/InternalRow;)V" of class
"org.apache.spark.sql.catalyst.expressions.GeneratedClass$SpecificUnsafeProjection" grows beyond 64 KB
```

Answer

If Project Tungsten is enabled, Spark will use codegen method to generate Java code for part of execution plan. However, each function in Java code to be compiled by JDK must be less than 64 KB. If complex SQL statements are run, the function in the Java code generated by codegen may exceed 64 KB, causing compilation failure.

To solve the problem, go to the **spark-defaults.conf** file on the client and set the **spark.sql.codegen.wholeStage** parameter to **false** to disable Project Tungsten.

24.16.2.17 Why Is Memory Insufficient if 10 Terabytes of TPCDS Test Suites Are Consecutively Run in Beeline/JDBCServer Mode?

Question

When the driver memory is set to 10 GB and the 10 TB TPCDS test suites are continuously run in Beeline/JDBCServer mode, SQL statements fail to be executed due to insufficient driver memory. Why?

Answer

By default, 1000 UI data records of jobs and stages are reserved in the memory.

The function of overflowing UI data to disks has been added to optimize large clusters. The overflow condition is that the size of UI data in each stage reaches the minimum threshold 5 MB. If the number of tasks in each stage is small, the size of UI data in the stage may not reach the threshold. As a result, the UI data in the stage is cached in the memory until the number of UI data records reaches the upper limit (1000 by default). Only then the old UI data is cleared from the memory.

Therefore, before the old UI data is cleared, the UI data occupies a large amount of memory. As a result, the driver memory is insufficient when 10 terabytes of TPCDS test suites are executed.

Workaround:

- Set **spark.ui.retainedJobs** and **spark.ui.retainedStages** based on service requirements to specify the number of UI data records of jobs and stages to be reserved. For details, see **Table 24-50** in **Spark Common Configuration Parameters**.
- If a large amount of UI data of jobs and stages needs to be reserved, increase the memory of the driver by setting the **spark.driver.memory** parameter. For details, see **Table 24-47** in **Spark Common Configuration Parameters**.

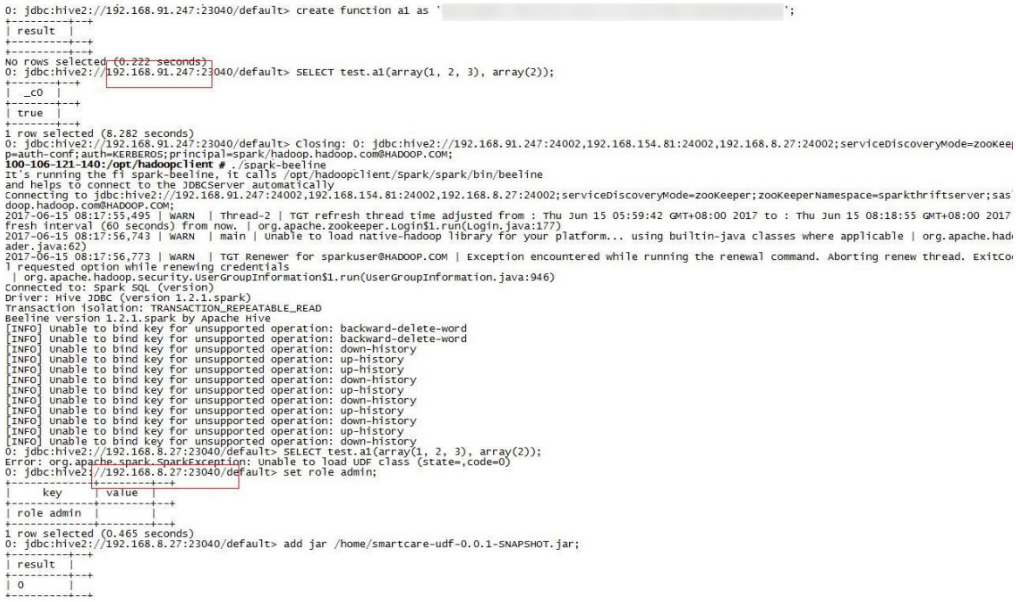
24.16.2.18 Why Functions Cannot Be Used When Different JDBCServers Are Connected?

Question

Scenario 1:

I run the **add jar** command to create permanent functions. When Beeline connects to different JDBCServers or the JDBCServer is restarted, I need to run the **add jar** command again.

Figure 24-16 Error information in Scenario 1



Scenario 2:

The functions can be queried by running the **show functions** command, but these functions cannot be used. This is because JAR files in the corresponding path do not exist on the connected JDBC node. After the JAR files are added, the query succeeds.

Figure 24-17 Error information in scenario 2

Answer

Scenario 1:

The **addjar** statement loads the jar only to the jarClassLoader of the currently connected JDBCServer. Different JDBCServers do not share the jarClassLoader.

After JDBCServer restarts, new jarClassLoader is created. So the **addjar** statement needs to be run again.

You can add a JAR file in either of the following ways: Add a JAR file when starting Spark SQL, for example, by running **spark-sql --jars /opt/test/two_udfs.jar**. Add a JAR file after Spark SQL is started, for example, by running **add jar /opt/test/two_udfs.jar**. The path specified by **add jar** can be a local or an HDFS path.

Scenario 2:

The **show functions** command obtains all functions in the current database from the external catalog. When a function is used in SQL, JDBCServer loads the JAR file corresponding to the function.

If the JAR file does not exist, the function cannot be used. In this case, run the ***add jar*** command again.

24.16.2.19 Why Does an Exception Occur When I Drop Functions Created Using the Add Jar Statement?

Question

- Question 1

  Why can I successfully drop functions without the drop function permission? The specific operations are as follows:

  a. On FusionInsight Manager, I added user **user1** and granted the user the admin permission.
  ```
  set role admin;add jar /home/smartcare-udf-0.0.1-SNAPSHOT.jar;create database db4;use db4;create function f11 as 'com.huaweixxx.smartcare.dac.hive.udf.UDFArrayGreaterEqual';create function f12 as 'com.huaweixxx.smartcare.dac.hive.udf.UDFArrayGreaterEqual';
  ```

  b. Then I canceled the admin permission:
  ```
  drop functiondb4.f11;
  ```

  The result shows that the drop operation is successful.

  Figure 24-18 Command output of the drop function command

  ```
  source /opt/${clientPath}/bigdata_env;/opt/${clientPath}/Spark2x/spark/bin/beeline -u 'jdbc:hive2://10.90.46.60:24002,10.90.46.61:24002,10.90.46.62:24002;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=sparkthriftserver2x;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.hadoop.com@HADOOP.COM;' -e "drop function db4.f11;;"
  ```

- Question 2

  After the preceding operation, I ran the **show function** command and the command output indicates that the function still exits. The specific operations are as follows:

  a. On FusionInsight Manager, I added user **user1** and granted the user the admin permission. Then I ran the following commands in spark-beeline:
  ```
  set role admin;create database db2;use db2;add jar /home/smartcare-udf-0.0.1-SNAPSHOT.jar;create function f11 as 'com.huaweixxx.smartcare.dac.hive.udf.UDFArrayGreaterEqual';create function f12 as 'com.huaweixxx.smartcare.dac.hive.udf.UDFArrayGreaterEqual';
  ```

  b. I exited and re-logged in to spark-beeline and ran the following command:
  ```
  set role admin;use db2;drop function db2.f11;
  ```

  c. I exited and re-logged in to spark-beeline and ran the following command:


```
use db2;show functions;
```

The result shows that the dropped function still exists.

Figure 24-19 Output of the show functions command

Answer

- Root cause:

  The root cause is that, in multi-active instance or multi-tenant mode, the function created using the **add jar** command in spark-beeline is invisible to JDBCServer instances. When you run the **drop function** command, if the JDBCServer instance connected by the session is not the JDBCServer instance that creates the function, the function cannot be found in the session. In addition, the default value of **hive.exec.drop.ignorenonexistent** in Hive is **true**. Therefore, if the function does not exist, no exception is reported when the function deletion operation is performed. As a result, no exception is reported when you delete a non-existing function and perform drop operation even though you do not have the drop function permission. When you run the **show function** command after restarting the session and connecting to the JDBCServer instance that creates the function, the command output indicates that the function still exists. This problem is inherited from the Hive community.

- Solution:

  Before you run the **drop function** command, run the **add jar** command. In this way, the drop operation succeeds only when you are granted the drop function permission, and the dropped function will not be displayed if you run the **show function** command.

24.16.2.20 Why Does Spark2x Have No Access to DataSource Tables Created by Spark1.5?

Question

When Spark2x accesses the DataSource table created by Spark1.5, a message is displayed indicating that schema information cannot be obtained. As a result, the table cannot be accessed. Why?

Answer

- Cause analysis:

  This is because the formats of the DataSource table information stored in Spark2x and Spark1.5 are inconsistent. Spark 1.5 divides schema information

into multiple parts and uses **path.park.0** as the key for storage. Spark 1.5 reads information from each part and reassembles the information into complete one. Spark2x directly uses the corresponding key to obtain the corresponding information. In this case, when Spark2x reads the DataSource table created by Spark1.5, the information corresponding to the key cannot be read. As a result, the DataSource table information fails to be parsed.

When processing Hive tables, Spark2x and Spark1.5 use the same storage mode. Therefore, Spark2x can directly read tables created by Spark1.5.

- Workaround:

  In Spark2x, create a foreign table to point to the actual data in the Spark1.5 table. In this way, the DataSource table created by Spark1.5 can be read in Spark2x. In addition, after Spark1.5 updates data, Spark2x can detect the change. The reverse is also true. In this way, Spark2x can access the DataSource table created by Spark1.5.

24.16.2.21 Why Cannot I Query Newly Inserted Data in an ORC Hive Table Using Spark SQL?

Question

Why cannot I query newly inserted data in an ORC Hive table using Spark SQL? This problem occurs in the following scenarios:

- For partitioned tables and non-partitioned tables, after data is inserted on the Hive client, the latest inserted data cannot be queried using Spark SQL.
- After data is inserted into a partitioned table using Spark SQL, if the partition information remains unchanged, the newly inserted data cannot be queried using Spark SQL.

Answer

To improve Spark performance, ORC metadata is cached. When the ORC table is updated by Hive or another means, the cached metadata remains unchanged, resulting in Spark SQL failing to query the newly inserted data.

For an ORC Hive partition table, if the partition information remains unchanged after data is inserted, the cached metadata is not updated. As a result, the newly inserted data cannot be queried by Spark SQL.

Solution

1. To solve the query problem, update metadata before starting a Spark SQL query.

   REFRESH TABLE *table_name*;

   table_name indicates the name of the table to be updated. The table must exist. Otherwise, an error is reported.

   When the query statement is executed, the latest inserted data can be obtained.

2. Run the following command to disable Spark optimization when using Spark:

   set spark.sql.hive.convertMetastoreOrc=false;

24.16.3 Spark Streaming

24.16.3.1 Same DAG Log Is Recorded Twice for a Streaming Task

Question

I use Spark Streaming to run the following command:

```
spark-submit -master yarn-client --conf spark.logLineage=true --jars $SPARK_HOME/jars/streamingClient/
kafka-clients-0.8.2.1.jar,$SPARK_HOME/jars/streamingClient/kafka_2.11-0.8.2.1.jar,$SPARK_HOME/jars/
streamingClient/spark-streaming-kafka-0-8_2.11-2.1.0.jar --class
com.huaweixxx.bigdata.spark.examples.FemaleInfoCollectionPrint /opt/female/
SparkStreamingJavaExample-1.0.jar <checkpoint> <batchTime> <windowTime> <topics> <brokers>
```

When there is no Kafka data input, the directed acyclic graph (DAG) of RDD displayed in the log is printed twice in a batch. The log is as follows:

```
-------------------------------------------
Time: 1491447950000 ms

-------------------------------------------
17/04/06 11:06:00 INFO SparkContext: RDD's recursive dependencies:
(2) MapPartitionsRDD[49] at filter at FemaleInfoCollectionPrint.java:111 []
  |  MapPartitionsRDD[48] at reduceByKeyAndWindow at FemaleInfoCollectionPrint.java:98 []
  |  CoGroupedRDD[47] at reduceByKeyAndWindow at FemaleInfoCollectionPrint.java:98 []
  |  MapPartitionsRDD[38] at reduceByKeyAndWindow at FemaleInfoCollectionPrint.java:98 []
  |      CachedPartitions: 2; MemorySize: 8.0 B; ExternalBlockStoreSize: 0.0 B; DiskSize: 0.0 B
  |  ReliableCheckpointRDD[40] at print at FemaleInfoCollectionPrint.java:123 []
  |  ShuffledRDD[36] at reduceByKeyAndWindow at FemaleInfoCollectionPrint.java:98 []
  |      CachedPartitions: 2; MemorySize: 8.0 B; ExternalBlockStoreSize: 0.0 B; DiskSize: 0.0 B
  +-(5) MapPartitionsRDD[35] at map at FemaleInfoCollectionPrint.java:81 []
     |  MapPartitionsRDD[34] at filter at FemaleInfoCollectionPrint.java:81 []
     |  MapPartitionsRDD[33] at map at FemaleInfoCollectionPrint.java:72 []
     |  MapPartitionsRDD[32] at map at FemaleInfoCollectionPrint.java:63 []
     |  KafkaRDD[31] at createDirectStream at FemaleInfoCollectionPrint.java:63 []
  |  ShuffledRDD[46] at reduceByKeyAndWindow at FemaleInfoCollectionPrint.java:98 []
  +-(5) MapPartitionsRDD[45] at map at FemaleInfoCollectionPrint.java:81 []
     |  MapPartitionsRDD[44] at filter at FemaleInfoCollectionPrint.java:81 []
     |  MapPartitionsRDD[43] at map at FemaleInfoCollectionPrint.java:72 []
     |  MapPartitionsRDD[42] at map at FemaleInfoCollectionPrint.java:63 []
  |  KafkaRDD[41] at createDirectStream at FemaleInfoCollectionPrint.java:63 []
17/04/06 11:06:00 INFO SparkContext: RDD's recursive dependencies:  (2) MapPartitionsRDD[48] at
reduceByKeyAndWindow at FemaleInfoCollectionPrint.java:98 [Memory Serialized 1x Replicated]
  |       CachedPartitions: 1; MemorySize: 4.0 B; ExternalBlockStoreSize: 0.0 B; DiskSize: 0.0 B
  |  CoGroupedRDD[47] at reduceByKeyAndWindow at FemaleInfoCollectionPrint.java:98 [Memory
Serialized 1x Replicated]
  |  MapPartitionsRDD[38] at reduceByKeyAndWindow at FemaleInfoCollectionPrint.java:98 [Memory
Serialized 1x Replicated]
  |      CachedPartitions: 2; MemorySize: 8.0 B; ExternalBlockStoreSize: 0.0 B; DiskSize: 0.0 B
  |  ReliableCheckpointRDD[40] at print at FemaleInfoCollectionPrint.java:123 [Memory Serialized 1x
Replicated]
  |  ShuffledRDD[36] at reduceByKeyAndWindow at FemaleInfoCollectionPrint.java:98 [Memory Serialized
1x Replicated]
  |      CachedPartitions: 2; MemorySize: 8.0 B; ExternalBlockStoreSize: 0.0 B; DiskSize: 0.0 B
  +-(5) MapPartitionsRDD[35] at map at FemaleInfoCollectionPrint.java:81 [Memory Serialized 1x
Replicated]
     |  MapPartitionsRDD[34] at filter at FemaleInfoCollectionPrint.java:81 [Memory Serialized 1x Replicated]
     |  MapPartitionsRDD[33] at map at FemaleInfoCollectionPrint.java:72 [Memory Serialized 1x Replicated]
     |  MapPartitionsRDD[32] at map at FemaleInfoCollectionPrint.java:63 [Memory Serialized 1x Replicated]
     |  KafkaRDD[31] at createDirectStream at FemaleInfoCollectionPrint.java:63 [Memory Serialized 1x
Replicated]
  |  ShuffledRDD[46] at reduceByKeyAndWindow at FemaleInfoCollectionPrint.java:98 [Memory Serialized
1x Replicated]
  |      CachedPartitions: 1; MemorySize: 4.0 B; ExternalBlockStoreSize: 0.0 B; DiskSize: 0.0 B
  +-(5) MapPartitionsRDD[45] at map at FemaleInfoCollectionPrint.java:81 [Memory Serialized 1x
Replicated]
```



```
    |  MapPartitionsRDD[44] at filter at FemaleInfoCollectionPrint.java:81 [Memory Serialized 1x Replicated]
    |  MapPartitionsRDD[43] at map at FemaleInfoCollectionPrint.java:72 [Memory Serialized 1x Replicated]
    |  MapPartitionsRDD[42] at map at FemaleInfoCollectionPrint.java:63 [Memory Serialized 1x Replicated]
 |  KafkaRDD[41] at createDirectStream at FemaleInfoCollectionPrint.java:63 [Memory Serialized 1x
Replicated]
-------------------------------------------
Time: 1491447960000 ms
-------------------------------------------
```

Answer

In this program, the print operator in DStream is used to display the result. This operator calls the take operator in RDD to implement bottom-layer calculation.

The take operator triggers calculation for multiple times by partition.

In this problem, due to the shuffle operation, the take operator has two partitions by default. Spark Streaming first calculates the first partition. But because there is no data input, the number of obtained results is less than 10. Then the second calculation is triggered. Therefore, the DAG of RDD is printed twice.

To resolve this issue, change the print operator to foreach(collect) in the code.

24.16.3.2 What Can I Do If Spark Streaming Tasks Are Blocked?

Question

After a Spark Streaming task is run and data is input, no processing result is displayed. Open the web page to view the Spark job execution status. The following figure shows that two jobs are waiting to be executed but cannot be executed successfully.

Figure 24-20 Active Jobs

Active Jobs (2)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total
3	print at test2StreamFromKafka.scala:31	2015/05/25 18:28:55	63.7 h	0/3
2	start at test2StreamFromKafka.scala:34	2015/05/25 18:28:55	63.7 h	0/1

Check the completed jobs. Only two jobs are found, indicating that Spark Streaming does not trigger data computing tasks. (By default, Spark Streaming has two jobs that attempt to run. See the figure below.)

Figure 24-21 Completed Jobs

Completed Jobs (2)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total
1	print at test2StreamFromKafka.scala:31	2015/05/25 18:28:55	0.7 s	2/2 (1 skipped)
0	start at test2StreamFromKafka.scala:34	2015/05/25 18:28:54	1 s	2/2

Answer

After fault locating, it is found that the number of computing cores of Spark Streaming is less than the number of receivers. As a result, after some receivers

are started, no resources are available to run computing tasks. Therefore, the first task keeps waiting and subsequent tasks keep queuing. **Figure 24-20** is an example of two queuing tasks.

To address this problem, it is advised to check whether the number of Spark cores is greater than the number of receivers when two tasks are queuing.

NOTE

Receiver is a permanent Spark job in Spark Streaming. It is common for Spark, but its life cycle is the same as that of a Spark Streaming task and occupies one computing core.

Pay attention to the relationship between the number of cores and the number of receivers in scenarios where default configurations are often used, such as debugging and testing.

24.16.3.3 What Should I Pay Attention to When Optimizing Spark Streaming Task Parameters?

Question

When Spark Streaming tasks are running, the data processing performance does not improve significantly as the number of executors increases. What should I pay attention to if I perform parameter optimization?

Answer

When the number of executor cores is 1, comply with the following rules to optimize Spark Streaming running parameters:

- The Spark task processing speed is related to the number of partitions in Kafka. When the number of partitions is less than the specified number of executors, the number of actually used executors is the same as the number of partitions, and other executors will be idle. Therefore, the number of executors must be less than or equal to the number of partitions.
- When data skew occurs on different partitions of Kafka, the executor corresponding to the partition with a large amount of data touches the glass ceiling of data processing. Therefore, when the Producer program is executed, data is sent to each partition on average to improve the processing speed.
- When partition data is evenly distributed, increasing the number of partitions and executors will improve the Spark processing speed. (When the number of partitions is the same as that of executors, the processing speed is the fastest.)
- When partition data is evenly distributed, ensure that the number of partitions is an integer multiple of the number of executors for proper allocation of resources.

24.16.3.4 Why Does the Spark Streaming Application Fail to Be Submitted After the Token Validity Period Expires?

Question

Change the validity period of the Kerberos ticket and HDFS token to 5 minutes, set **dfs.namenode.delegation.token.renew-interval** to a value less than 60 seconds,

and submit the Spark Streaming application. If the token expires, the error message below is displayed, and the application exits. Why?

```
token (HDFS_DELEGATION_TOKEN token 17410 for spark2x) is expired
```

Answer

- Possible causes:

  The credential refresh thread of the ApplicationMaster process uploads the updated credential file to the HDFS based on the *token renew period multiplied by 0.75*.

  In the executor process, the credential refresh thread obtains the updated credential file from the HDFS based on the time ratio of the *token renewal period multiplied by 0.8* to update the token in UserGroupInformation, preventing the token from being invalid.

  When the credential refresh thread of the executor process detects that the current time is later than the credential file update time (*token renew period x 0.8*), it waits for 1 minute and then obtains the latest credential file from the HDFS to ensure that the AM has stored the updated credential file in the HDFS.

  When the value of **dfs.namenode.delegation.token.renew-interval** is less than 60 seconds, the started executor detects that the current time is later than the time when the credential file is updated. One minute later, the executor obtains the latest credential file from the HDFS. However, the token is already invalid, and the task fails to be executed. Then, other executor processes retry within 1 minute. The task also fails to run on other executors. As a result, the executors that fail to run are added to the blacklist. If no executors are available, the application exits.

- Solution:

  In the Spark application scenario, set **dfs.namenode.delegation.token.renew-interval** to a value greater than 80 seconds. For details about the **dfs.namenode.delegation.token.renew-interval** parameter, see **Table 24-84**.

  Table 24-84 Parameter description

Parameter	Description	Default Value
dfs.namenode.delegation.token.renew-interval	This parameter is a server parameter. It specifies the maximum lifetime to renew a token. Unit: milliseconds.	86400000

24.16.3.5 Why Does the Spark Streaming Application Fail to Be Started from the Checkpoint When the Input Stream Has No Output Logic?

Question

One input stream was created for the Spark Streaming application, but the input stream had no output logic. The application failed to be started from the checkpoint. The error information is as follows:

```
17/04/24 10:13:57 ERROR Utils: Exception encountered
java.lang.NullPointerException
at org.apache.spark.streaming.dstream.DStreamCheckpointData$$anonfun$writeObject$1.apply$mcV
$sp(DStreamCheckpointData.scala:125)
at org.apache.spark.streaming.dstream.DStreamCheckpointData$$anonfun$writeObject
$1.apply(DStreamCheckpointData.scala:123)
at org.apache.spark.streaming.dstream.DStreamCheckpointData$$anonfun$writeObject
$1.apply(DStreamCheckpointData.scala:123)
at org.apache.spark.util.Utils$.tryOrIOException(Utils.scala:1195)
at
org.apache.spark.streaming.dstream.DStreamCheckpointData.writeObject(DStreamCheckpointData.scala:123
)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at java.io.ObjectStreamClass.invokeWriteObject(ObjectStreamClass.java:1028)
at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1496)
at java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432
at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
at java.io.ObjectOutputStream.defaultWriteFields(ObjectOutputStream.java:1548)
at java.io.ObjectOutputStream.defaultWriteObject(ObjectOutputStream.java:441)
at org.apache.spark.streaming.dstream.DStream$$anonfun$writeObject$1.apply$mcV$sp(DStream.scala:515)
at org.apache.spark.streaming.dstream.DStream$$anonfun$writeObject$1.apply(DStream.scala:510)
at org.apache.spark.streaming.dstream.DStream$$anonfun$writeObject$1.apply(DStream.scala:510)
at org.apache.spark.util.Utils$.tryOrIOException(Utils.scala:1195)
at org.apache.spark.streaming.dstream.DStream.writeObject(DStream.scala:510)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at java.io.ObjectStreamClass.invokeWriteObject(ObjectStreamClass.java:1028)
at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1496)
at java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432
at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
at java.io.ObjectOutputStream.writeArray(ObjectOutputStream.java:1378)
at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1174)
at java.io.ObjectOutputStream.defaultWriteFields(ObjectOutputStream.java:1548)
at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1509)
at java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432
at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
at java.io.ObjectOutputStream.defaultWriteFields(ObjectOutputStream.java:1548)
at java.io.ObjectOutputStream.defaultWriteObject(ObjectOutputStream.java:441)
at org.apache.spark.streaming.DStreamGraph$$anonfun$writeObject$1.apply$mcV
$sp(DStreamGraph.scala:191)
at org.apache.spark.streaming.DStreamGraph$$anonfun$writeObject$1.apply(DStreamGraph.scala:186)
at org.apache.spark.streaming.DStreamGraph$$anonfun$writeObject$1.apply(DStreamGraph.scala:186)
at org.apache.spark.util.Utils$.tryOrIOException(Utils.scala:1195)
at org.apache.spark.streaming.DStreamGraph.writeObject(DStreamGraph.scala:186
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at java.io.ObjectStreamClass.invokeWriteObject(ObjectStreamClass.java:1028)
at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1496)
at java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432
at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
at java.io.ObjectOutputStream.defaultWriteFields(ObjectOutputStream.java:1548)
```



```
at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1509)
at java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432
at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java:348)
at org.apache.spark.streaming.Checkpoint$$anonfun$serialize$1.apply$mcV$sp(Checkpoint.scala:142)
at org.apache.spark.streaming.Checkpoint$$anonfun$serialize$1.apply(Checkpoint.scala:142)
at org.apache.spark.streaming.Checkpoint$$anonfun$serialize$1.apply(Checkpoint.scala:142)
at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1230)
at org.apache.spark.streaming.Checkpoint$.serialize(Checkpoint.scala:143)
at org.apache.spark.streaming.StreamingContext.validate(StreamingContext.scala:566)
at org.apache.spark.streaming.StreamingContext.liftedTree1$1(StreamingContext.scala:612)
at org.apache.spark.streaming.StreamingContext.start(StreamingContext.scala:611)
at com.spark.test.kafka08LifoTwoInkfk$.main(kafka08LifoTwoInkfk.scala:21)
at com.spark.test.kafka08LifoTwoInkfk.main(kafka08LifoTwoInkfk.scala)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.apache.spark.deploy.SparkSubmit$.org$apache$spark$deploy$SparkSubmit$
$runMain(SparkSubmit.scala:772)
at org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala:183)
at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:208)
at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:123)
at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
```

Answer

When Streaming Context is started, if a checkpoint is set for an application, the DStream checkpoint object in the application needs to be serialized. **dstream.context** is used during serialization.

dstream.context is used to reversely search for dependent DStreams from output Streams when Streaming Context is started, and to set **context** one by one. If an input stream is created for a Spark Streaming application but the input stream has no output logic, no **context** is set for the input stream. As a result, **NullPointerException** is reported during serialization.

Solution: If the input stream of the output logic does not exist in the application, delete the input stream from the code or add the output logic of the input stream.

24.16.3.6 Why Is the Input Size Corresponding to Batch Time on the Web UI Set to 0 Records When Kafka Is Restarted During Spark Streaming Running?

Question

When the Kafka is restarted during the execution of the Spark Streaming application, the application cannot obtain the topic offset from the Kafka. As a result, the job fails to be generated. As shown in **Figure 24-22**, **2017/05/11 10:57:00-2017/05/11 10:58:00** indicates the Kafka restart time. After the restart is successful at 10:58:00 on May,11,2017, the value of **Input Size** is **0 records**.

Figure 24-22 On the Web UI, the **input size** corresponding to the **batch time** is **0 records**.

Completed Batches (last 9 out of 9)

Batch Time	Input Size	Scheduling Delay (?)	Processing Time (?)	Total Delay (?)	Output Ops: Succeeded/Total
2017/05/11 10:58:50	18 records	0 ms	0.4 s	0.4 s	1/1
2017/05/11 10:58:40	20 records	4 s	0.3 s	4 s	1/1
2017/05/11 10:58:30	20 records	14 s	0.5 s	14 s	1/1
2017/05/11 10:58:20	20 records	23 s	0.4 s	24 s	1/1
2017/05/11 10:58:10	20 records	33 s	0.5 s	33 s	1/1
2017/05/11 10:58:00	0 records	6 ms	43 s	43 s	1/1
2017/05/11 10:57:00	19 records	1 ms	0.9 s	0.9 s	1/1
2017/05/11 10:56:50	20 records	1 ms	0.6 s	0.6 s	1/1
2017/05/11 10:56:40	28 records	13 ms	5 s	5 s	1/1

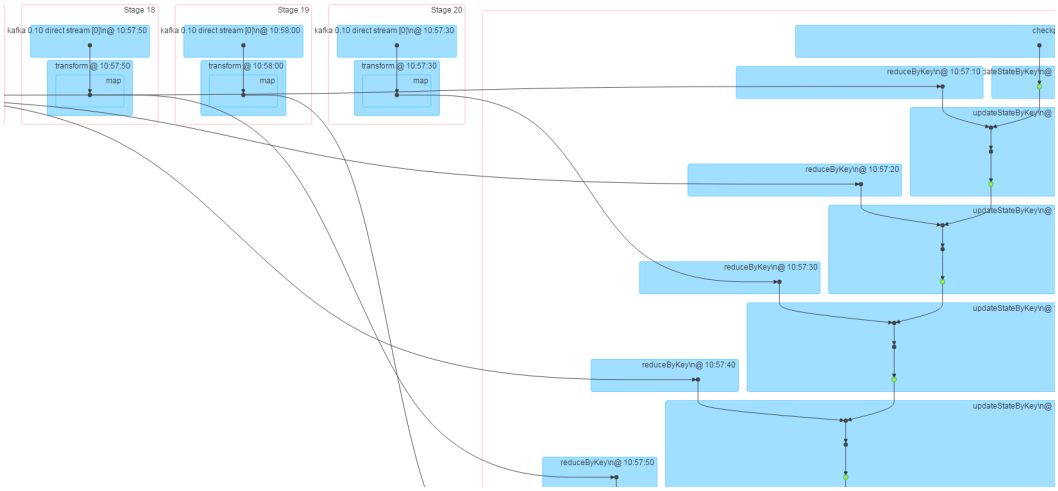
Answer

After Kafka is restarted, the application supplements the missing RDD between 10:57:00 on May 11, 2017 and 10:58:00 on May 11, 2017 based on the batch time. See **Figure 24-23**. Although the number of read data records displayed on the UI is **0**, the missing data is processed in the supplemented RDD. Therefore, no data loss occurs.

The data processing mechanism during the Kafka restart period is as follows:

The Spark Streaming application uses the **state** function (for example, **updateStateByKey**). After Kafka is restarted, the Spark Streaming application generates a batch task at 10:58:00 on May 11, 2017. The missing RDD between10:57:00 on May 11, 2017 and 10:58:00 on May 11, 2017 is supplemented based on the batch time (data that is not read in Kafka before Kafka restart, which belongs to the batch before 10:57:00 on May 11, 2017). See **Figure 24-23**.

Figure 24-23 Mechanism for processing missing data during the restart



24.16.4 What Should I Do If Recycle Bin Version I Set on the Spark Client Does Not Take Effect?

Question

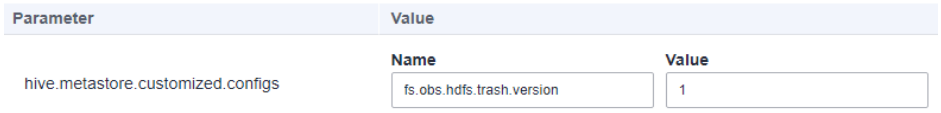
The setting of **fs.obs.hdfs.trash.version=1** on the Spark client did not take effect. After table was dropped, the path for storing files in the recycle bin remained unchanged.

Generally, the default settings are as follows:

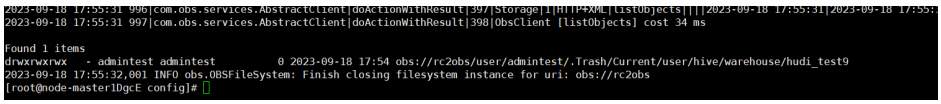
- If **fs.obs.hdfs.trash.version** is set to **2**, the recycle bin path is **/user/.Trash/${userName}/Current**.
- If **fs.obs.hdfs.trash.version** is set to **1**, the recycle bin path is **/user/${userName}/.Trash/Current**.

Answer

Log in to Manager and choose **Cluster** > **Services** > **Hive**, click **Configurations**, click **All Configurations**, and select **MetaStore (Role)** > **Customization**. On the displayed page, set **hive.metastore.customized.configs**. Set **fs.obs.hdfs.trash.version** to **1**, save the settings, and restart the Metastore instance.



After Hive Metastore is configured, the recycle bin path is correct.



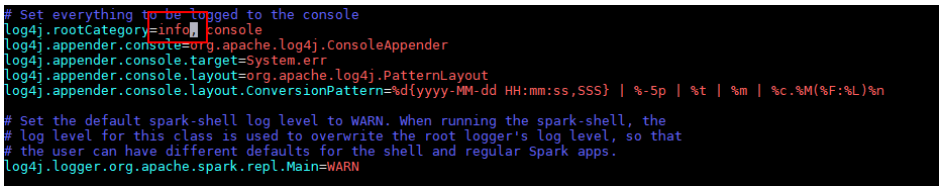
24.16.5 How Do I Change the Log Level to INFO When Using Spark yarn-client?

Question

How do I change the log level to INFO when using Spark yarn-client?

Answer

1. Log in to the Spark client node and change the value of **Log4j.rootCategory** in the {*Client installation directory*}**Spark/spark/conf/log4j.properties** configuration file to **INFO**.



2. Restart the spark-sql client.

24.17 Spark Troubleshooting

24.17.1 Why the Job Information Obtained from the restful Interface of an Ended Spark Application Is Incorrect?

Question

The job information obtained from the restful interface of an ended Spark application is incorrect: the value of **numActiveTasks** is negative, as shown in **Figure 24-24**:

Figure 24-24 job information


```
[ {
  "jobId" : 0,
  "name" : "reduce at SparkPi.scala:36",
  "submissionTime" : "2016-05-28T09:35:34.415GMT",
  "completionTime" : "2016-05-28T09:35:35.686GMT",
  "stageIds" : [ 0 ],
  "status" : "SUCCEEDED",
  "numTasks" : 2,
  "numActiveTasks" : -1,
  "numCompletedTasks" : 2,
  "numSkippedTasks" : 2,
  "numFailedTasks" : 0,
  "numActiveStages" : 0,
  "numCompletedStages" : 1,
  "numSkippedStages" : 0,
  "numFailedStages" : 0
} ]
```


NOTE

numActiveTasks indicates the number of active tasks.

Answer

The job information can be obtained in either of the following methods:

- Set **spark.history.briefInfo.gather=true** and then view the brief JobHistory information.
- Visit the JobHistory2x page of Spark (URL: https://IP:port/api/v1/<appid>/jobs/).

The value of **numActiveTasks** in the job information is calculated from the difference between the number of SparkListenerTaskStart events and the number of SparkListenerTaskEnd events in the **eventLog** file. If some events are not recorded in the **eventLog** file, the job information obtained from the restful interface is incorrect.

24.17.2 Why Cannot I Switch from the Yarn Web UI to the Spark Web UI?

Question

In FusionInsight, the Spark application is run in yarn-client mode on the client. The following error occurs during the switch from the Yarn web UI to the application web UI:

The YARN ResourceManager log shows the following information:

```
2016-07-21 16:35:27,099 | INFO  | Socket Reader #1 for port 8032 | Auth successful for mapred/
hadoop.<System domain name>@<System domain name> (auth:KERBEROS) | Server.java:1388
2016-07-21 16:35:27,105 | INFO  | 1526016381@qtp-1178290888-1015 | admin is accessing unchecked
http://10.120.169.53:23011 which is the app master GUI of
application_1468986660719_0045 owned by spark | WebAppProxyServlet.java:393
2016-07-21 16:36:02,843 | INFO  | Socket Reader #1 for port 8032 | Auth successful for hive/
hadoop.<System domain name>@<System domain name> (auth:KERBEROS) | Server.java:1388
2016-07-21 16:36:02,851 | INFO  | Socket Reader #1 for port 8032 | Auth successful for hive/
hadoop.<System domain name>@<System domain name> (auth:KERBEROS) | Server.java:1388
2016-07-21 16:36:12,163 | WARN  | 1526016381@qtp-1178290888-1015 | /proxy/
application_1468986660719_0045/: java.net.ConnectException: Connection timed out |
Slf4jLog.java:76
2016-07-21 16:37:03,918 | INFO  | Socket Reader #1 for port 8032 | Auth successful for hive/
hadoop.<System domain name>@<System domain name> (auth:KERBEROS) | Server.java:1388
2016-07-21 16:37:03,926 | INFO  | Socket Reader #1 for port 8032 | Auth successful for hive/
hadoop.<System domain name>@<System domain name> (auth:KERBEROS) | Server.java:1388
2016-07-21 16:37:11,956 | INFO  | AsyncDispatcher event handler | Updating application attempt
appattempt_1468986660719_0045_000001 with final state: FINISHING,
and exit status: -1000 | RMAppAttemptImpl.java:1253
```

Answer

On FusionInsight Manager, the IP address of the Yarn service is in the 192 network segment.

In Yarn logs, the IP address of Spark web UI read by Yarn is http://10.120.169.53:23011, which is in the 10 network segment. The IP addresses in the 192 network segment cannot communicate with those in the 10 network segment. As a result, the Spark web UI fails to be accessed.

Solution:

Log in to the client whose IP address is **10.120.169.53** and change the IP address in the **/etc/hosts** file to the IP address in the 192 network segment. Run the Spark application again. The Spark web UI is displayed.

24.17.3 What Can I Do If an Error Occurs when I Access the Application Page Because the Application Cached by HistoryServer Is Recycled?

Question

An error occurs when I access a Spark application page on the HistoryServer page.

Check the HistoryServer logs. The "FileNotFound" exception is found. The related logs are as follows:

```
2016-11-22 23:58:03,694 | WARN  | [qtp55429210-232] | /history/application_1479662594976_0001/stages/stage/ | org.sparkproject.jetty.servlet.ServletHandler.doHandle(ServletHandler.java:628)
java.io.FileNotFoundException: ${BIGDATA_HOME}/tmp/spark/jobHistoryTemp/blockmgr-5f1f6aca-2303-4290-9845-88fa94d78480/09/temp_shuffle_11f82aaf-e226-46dc-b1f0-002751557694 (No such file or directory)
```

Answer

If a Spark application with a large number of tasks is run on the HistoryServer page, the memory overflows to disk and files with the **temp_shuffle** prefix are generated.

By default, HistoryServer caches 50 Spark applications (determined by the **spark.history.retainedApplications** configuration item). When the number of Spark applications in the memory exceeds 50, HistoryServer reclaims the first cached Spark application and clears the corresponding **temp_shuffle** file.

When a user is viewing Spark applications to be recycled, the **temp_shuffle** file may not be found. As a result, the current page cannot be accessed.

If the preceding problem occurs, use either of the following methods to solve the problem:

- Access the HistoryServer page of the Spark application again. The correct page information is displayed.
- If more than 50 Spark applications need to be accessed at the same time, increase the value of **spark.history.retainedApplications**.

  Log in to FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Service** > **Spark2x** > **Configuration**, and click **All Configurations**. In the navigation tree on the left, choose **JobHistory2x** > **GUI**, and set parameters.

  Table 24-85 Parameter description

Parameter	Description	Default Value
spark.history.retainedApplications	Number of Spark applications cached by HistoryServer. When the number of applications to be cached exceeds the value of this parameter, HistoryServer reclaims the first cached Spark application.	50

24.17.4 Apps Cannot Be Displayed on the JobHistory Page When an Empty Part File Is Loaded

Question

If an application is executed in group mode and the corresponding **part** file in HDFS is empty, the app will not be displayed on the JobHistory home page.

Answer

When updating an app on the page, the JobHistory service checks the size of the part file in HDFS to determine whether to update the app information displayed on the home page. If the file has not been viewed before, the system compares its size with 0. If the size is greater than 0, the system reads the file.

In the case of grouping, if no job of the executed app is currently running, the **part** file will be empty. In this case, the JobHistory service will not read the file, and the app will not be displayed on the JobHistory page. However, if the size of the **part** file is updated later, JobHistory will display the app again.

24.17.5 Why Does Spark Fail to Export a Table with the Same Field Name?

Question

The following code fails to be executed on spark-shell of Spark:

```
val acctId = List(("49562", "Amal", "Derry"), ("00000", "Fred", "Xanadu"))
val rddLeft = sc.makeRDD(acctId)
val dfLeft = rddLeft.toDF("Id", "Name", "City")
//dfLeft.show
val acctCustId = List(("Amal", "49562", "CO"), ("Dave", "99999", "ZZ"))
val rddRight = sc.makeRDD(acctCustId)
val dfRight = rddRight.toDF("Name", "CustId", "State")
//dfRight.show
val dfJoin = dfLeft.join(dfRight, dfLeft("Id") === dfRight("CustId"), "outer")
dfJoin.show
dfJoin.repartition(1).write.format("com.databricks.spark.csv").option("delimiter", "\t").option("header",
"true").option("treatEmptyValuesAsNulls", "true").option("nullValue", "").save("/tmp/outputDir")
```

Answer

When Spark exports tables with the same field name, the export fails.

In Spark, the duplicate field name of the **join** statement is checked. You need to modify the code to ensure that no duplicate field exists in the saved data.

24.17.6 Why JRE fatal error after running Spark application multiple times?

Question

Why JRE fatal error after running Spark application multiple times?

Answer

When you run Spark application multiple times, JRE fatal error occurs and this is due to the problem with the Linux Kernel.

To resolve this issue, upgrade the **kernel version to 4.13.9-2.ge7d7106-default**.

24.17.7 Native Spark2x UI Fails to Be Accessed or Is Incorrectly Displayed when Internet Explorer Is Used for Access

Question

When Internet Explorer 9, 10, or 11 is used to access the native Spark2x UI, the access fails or the page is incorrectly displayed.

Symptom

Internet Explorer fails to access the native Spark2x UI.

Cause

Some versions of Internet Explorer 9, 10, and 11 fail to process SSL handshakes.

Solution

Use Google Chrome 71 or later for access.

24.17.8 How Does Spark2x Access External Cluster Components?

Question

How can I use Spark2x in cluster 1 to connect with HDFS, Hive, HBase, and Kafka components in cluster 2?

Answer

1. Components in two clusters can access each other. However, there are the following restrictions:
   - Only one Hive MetaStore can be accessed. Specifically, Hive MetaStore in cluster 1 and Hive MetaStore in cluster 2 cannot be accessed at the same time.
   - User systems in different clusters are not synchronized. When users access components in another cluster, user permission is determined by the user configuration of the peer cluster. For example, if user A of cluster

1 does not have the permissions to access the HBase meta table in cluster 1 but user A of cluster 2 can access the HBase meta table in cluster 2, user A of cluster 1 can access the HBase meta table in cluster 2.

- To enable components in a security cluster to communicate with each other across Manager, you need to configure mutual trust.

2. The following describes how to access Hive, HBase, and Kafka components in cluster 2 as user A.

   NOTE

   The following operations are based on the scenario where a user uses the FusionInsight client to submit the Spark2x application. If the user uses the configuration file directory, the user needs to modify the corresponding file in the configuration directory of the application and upload the configuration file to the executor.

   When the HDFS and HBase clients access the server, **hostname** is used to configure the server address. Therefore, the hosts configuration of all nodes to be accessed must be saved in the **/etc/hosts** file on the client. You can add the host of the peer cluster node to the **/etc/hosts** file of the client node in advance.

   - Access Hive metastore: Replace the **hive-site.xml** file in the **conf** directory of the Spark2x client in cluster 1 with the **hive-site.xml** file in the **conf** directory of the Spark2x client in cluster 2.

     After the preceding operations are performed, you can use Spark SQL to access Hive MetaStore. To access Hive table data, you need to perform the operations in **• Access HDFS of two clusters at the same time:** and set **nameservice** of the peer cluster to **LOCATION**.

   - Access HBase of the peer cluster.

     i. Configure the IP addresses and host names of all ZooKeeper nodes and HBase nodes in cluster 2 in the **/etc/hosts** file on the client node of cluster 1.

     ii. Replace the **hbase-site.xml** file in the **conf** directory of the Spark2x client in cluster 1 with the **hbase-site.xml** file in the **conf** directory of the Spark2x client in cluster 2.

   - Access Kafka: Set the address of the Kafka Broker to be accessed to the Kafka Broker address in cluster 2.

   - Access HDFS of two clusters at the same time:

     - Two tokens with the same NameService cannot be obtained. The NameServices of the HDFS in two clusters must be different. For example, one is **hacluster**, and the other is **test**.

       1) Obtain the following configurations from the **hdfs-site.xml** file of cluster2 and add them to the **hdfs-site.xml** file in the **conf** directory of the Spark2x client in cluster1:

          dfs.nameservices.mappings, **dfs.nameservices**, **dfs.namenode.rpc-address.test.***, **dfs.ha.namenodes.test**, and **dfs.client.failover.proxy.provider.test**

          The following is an example:

          ```
          <property>
          <name>dfs.nameservices.mappings</name>
          <value>[{"name":"hacluster","roleInstances":["14","15"]},
          {"name":"test","roleInstances":["16","17"]}]</value>
          </property>
          ```

```
<property>
<name>dfs.nameservices</name>
<value>hacluster,test</value>
</property>
<property>
<name>dfs.namenode.rpc-address.test.16</name>
<value>192.168.0.1:8020</value>
</property>
<property>
<name>dfs.namenode.rpc-address.test.17</name>
<value>192.168.0.2:8020</value>
</property>
<property>
<name>dfs.ha.namenodes.test</name>
<value>16,17</value>
</property>
<property>
<name>dfs.client.failover.proxy.provider.test</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider
</value>
</property>
```

2) Modify **spark.yarn.extra.hadoopFileSystems = hdfs://test and spark.hadoop.hdfs.externalToken.enable = true** in the **spark-defaults.conf** configuration file under the **conf** directory on the Spark client of cluster 1.

```
spark.yarn.extra.hadoopFileSystems = hdfs://test
spark.hadoop.hdfs.externalToken.enable = true
```

3) In the application submission command, add the **--keytab** and **--principal** parameters and set them to the user who submits the task in cluster1.

4) Use the Spark client of cluster1 to submit the application. Then, both HDFS services can be accessed.

- Access HBase of two clusters at the same time:

  i. Modify **spark.hadoop.hbase.externalToken.enable = true** in the **spark-defaults.conf** configuration file under the **conf** directory on the Spark client of cluster 1.

  ```
  spark.hadoop.hbase.externalToken.enable = true
  ```

  ii. When accessing HBase, you need to use the configuration file of the corresponding cluster to create a **Configuration** object for creating a **Connection** object.

  iii. In an MRS cluster, tokens of multiple HBase services can be obtained at the same time to solve the problem that the executor cannot access HBase. The method is as follows:

  Assume that you need to access HBase of the current cluster and HBase of cluster2. Save the **hbase-site.xml** file of cluster2 in a compressed package named **external_hbase_conf***, and use **--archives** to specify the compressed package when submitting the command.

24.17.9 Why Does the Foreign Table Query Fail When Multiple Foreign Tables Are Created in the Same Directory?

Question

Assume there is a data file path named **/test_data_path**. User A creates a foreign table named **tableA** for the directory, and user B creates a foreign table named

tableB for the directory. When user B performs the insert operation on **tableB**, user A fails to query data using **tableA** and the error "Permission denied" is displayed.

Answer

After user B performs the insert operation on **tableB**, a new data file is generated in the foreign table path and the file belongs to user B. When user A queries data using **tableA**, all files in the foreign table directory are read. In this case, the query fails because user A does not have the read permissions on the file generated by user B.

This problem also occurs in other scenarios. For example, the **inset overwrite** operation will also duplicate other table files in this directory.

Due to the Spark SQL implementation mechanism, check restrictions in this scenario will lead to inconsistency and performance deterioration. Therefore, no restriction is added in this scenario, and this method is not recommended.

24.17.10 Why Is the Native Page of an Application in Spark2x JobHistory Displayed Incorrectly?

Question

Submit a Spark application that contains millions of tasks in a single job. If you try to access the native page of an application in JobHistory after it has ended, the browser may take a long time to switch to the page. If the switch does not happen within 10 minutes, "Proxy Error" will be displayed.

Figure 24-25 Example error information

Answer

When you switch to the native page of an application in JobHistory, JobHistory needs to replay the event logs of the application. If the application contains a large number of event logs, the replay takes a long time and the browser waits for a long time.

When the browser accesses the native page in JobHistory, the httpd proxy is required. The timeout interval of the proxy is 10 minutes. Therefore, if JobHistory cannot parse and return the event logs within 10 minutes, httpd returns the "Proxy Error" message to the browser.

Solution

The local disk caching function is enabled for JobHistory. When an application is accessed, the parsing result of the event logs of the application is cached to the local disk. When the application is accessed for the second time, the response

speed is significantly improved. In this case, you only need to wait for a while and access the original link again.

24.17.11 Why Do I Fail to Create a Table in the Specified Location on OBS After Logging to spark-beeline?

Question

When the OBS ECS/BMS image cluster is connected, after spark-beeline is logged in, an error is reported when a location is specified to create a table on OBS.

Figure 24-26 Error message

```
de-master2qCKJ:22550/> create database sparkdb location 'obs://800mrs/sparktest/sparkdb';



0.626 seconds)
de-master2qCKJ:22550/> use sparkdb;


0.072 seconds)
de-master2qCKJ:22550/> create table orc (id int,name string) using orc;
xception: Configuration problem with provider path. (state=,code=0)
```

Answer

The permission on the **ssl.jceks** file in HDFS is insufficient. As a result, the table fails to be created.

```
Caused by: org.apache.hadoop.security.AccessControlException: Permission denied: user=root, access=READ, inode="/user/spark2x/jars/8.0.2/ssl.jceks":spark2x:hadoop:-rw-------
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:410)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:264)
    at com.[redacted].hadoop.adapter.hdfs.plugin.HWAccessControlEnforce.checkPermission([redacted].java:54)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:194)
    at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermission(FSDirectory.java:1957)
    at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermission(FSDirectory.java:1941)
    at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPathAccess(FSDirectory.java:1891)
    at org.apache.hadoop.hdfs.server.namenode.FSDirStatAndListingOp.getBlockLocations(FSDirStatAndListingOp.java:175)
    at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getBlockLocations(FSNamesystem.java:1990)
    at org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.getBlockLocations(NameNodeRpcServer.java:762)
    at org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTranslatorPB.getBlockLocations(ClientNamenodeProtocolServerSideTranslatorPB.java:445)
    at org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$ClientNamenodeProtocol$2.callBlockingMethod(ClientNamenodeProtocolProtos.java)
    at org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.call(ProtobufRpcEngine.java:528)
    at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:1036)
    at org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:985)
    at org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:913)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:422)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1737)
    at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2876)
```

Solution

1. Log in to the node where Spark2x resides as user **omm** and run the following command:

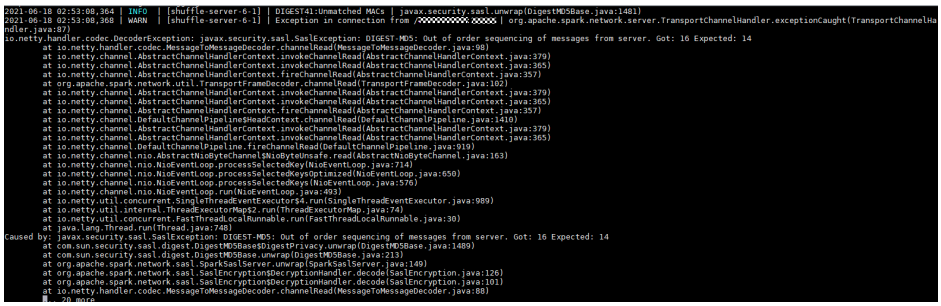
   vi ${BIGDATA_HOME}/FusionInsight_Spark2x_xxx/install/FusionInsight-Spark2x-*/spark/sbin/fake_prestart.sh

2. Change **eval "${hdfsCmd}" -chmod 600 "${InnerHdfsDir}"/ssl.jceks >> "${PRESTART_LOG}" 2>&1** to **eval "${hdfsCmd}" -chmod 644 "${InnerHdfsDir}"/ssl.jceks >> "${PRESTART_LOG}" 2>&1**.
3. Restart the SparkResource instance.

24.17.12 Spark Shuffle Exception Handling

Question

In some scenarios, the following exception occurs in the Spark shuffle phase:



Solution

For JDBC:

Log in to FusionInsight Manager, change the value of the JDBCServer parameter **spark.authenticate.enableSaslEncryption** to **false**, and restart the corresponding instance.

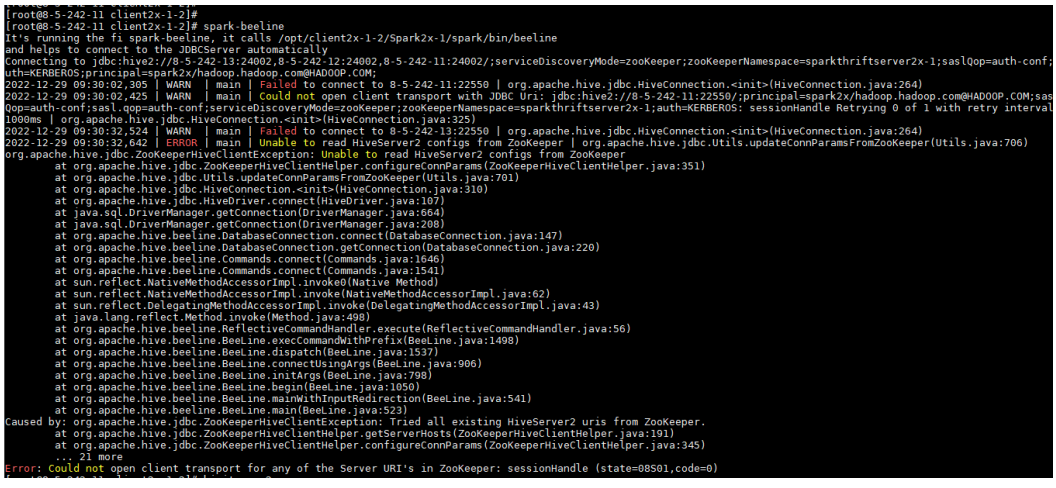
For client jobs:

When the client submits the application, change the value of **spark.authenticate.enableSaslEncryption** in the **spark-defaults.conf** file to **false**.

24.17.13 Why Cannot Common Users Log In to the Spark Client When There Are Multiple Service Scenarios in Spark?

Question

If multiple service scenarios are used in Spark, common users may not be able to log in to Spark-Beeline. The error information shown in the following figure is displayed.



Cause

If there is a multi-scenario service in Hive, common users who are not part of the Hive user group and do not have permissions on the Hive directory will not be able to log in.

Solution

Log in to FusionInsight Manager, change the user group to which common users belong, and add common users to all user groups in Hive.

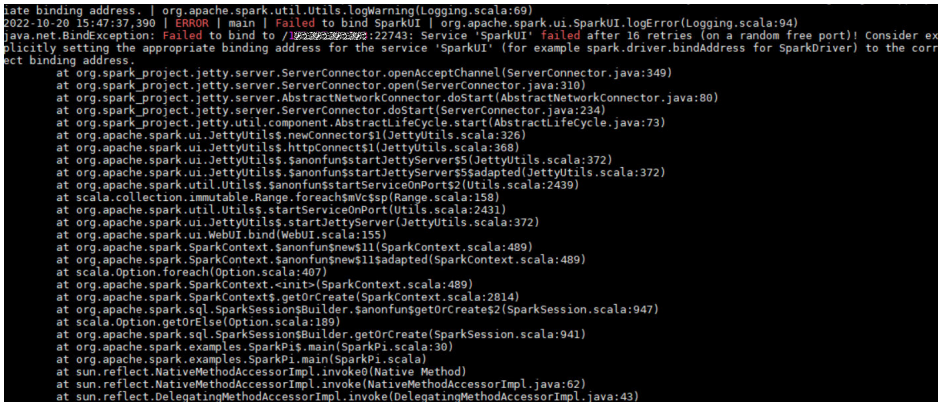
24.17.14 Why Does the Cluster Port Fail to Connect When a Client Outside the Cluster Is Installed or Used?

Question

When a client outside the cluster is installed or used, the Spark task port sometimes fails to be connected.

Exception information: "Failed to bind SparkUi"

```
Cannot assign requested address: Service 'sparkDriver' failed after 16 retries (on a random free port)! Consider explicitly setting the appropriate binding address for the service 'sparkDriver' (for example spark.driver.bindAddress for SparkDriver) to the correct binding address.
```



Causes

- The network between the cluster node and the client node is disconnected.
- The firewall on the client node is not disabled.
- If the port is occupied, each Spark task occupies a SparkUI port. The default port number is **22600**. If the port is occupied, increase the port number in sequence and try again. However, there are only 16 retries by default. After the 16 retries, this task is aborted.
- The Spark configuration parameters on the client are incorrect.
- The code is incorrect.

Solution

The application cannot access the IP address and port number of SparkUI. You can follow the steps below to identify potential causes and try to solve this problem:

- Verify if the cluster node can communicate with the client node.

  Run the following command on the client node to check whether the cluster node mapping is configured in the **/etc/hosts** file on the client node:

  ping *SparkUI IP address*

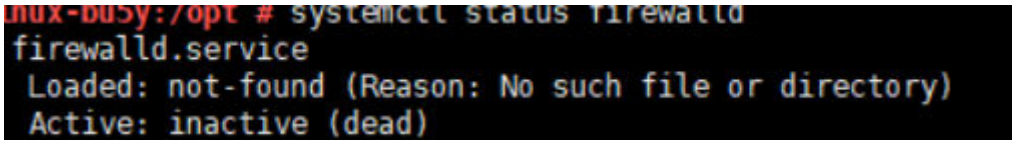
If the IP address cannot be pinged, check the mapping and network configurations.

- Disable the firewall on the client node.

  Run the following command to check whether the function is disabled:

  systemctl status firewalld (The query command varies depending on the OS. This command uses CentOS as an example.)

  As shown in the following figure, **dead** indicates that the function is disabled.

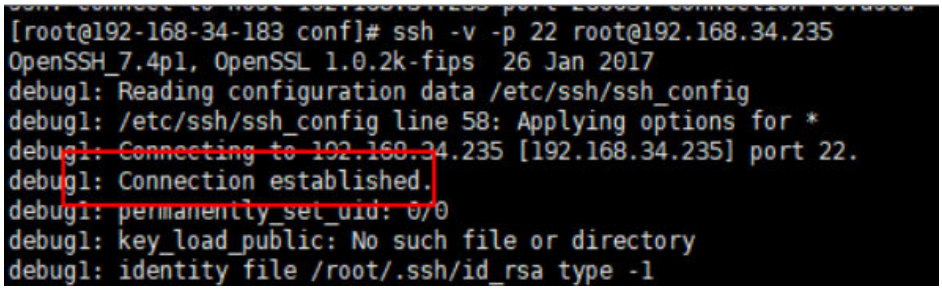
  If the firewall is enabled, the communication is affected. Run the following command to disable the firewall:

  service firewalld stop (The query command varies depending on the OS. This command uses CentOS as an example.)

- Check whether the port is occupied.

  ssh -v -p port username@ip

  If the message "Connection established" appears, it means that the connection was successful and the port is in use.

  The Spark UI port range is determined by the **spark.random.port.min** and **spark.random.port.max** parameters in the **spark-defaults.conf** configuration file. If all ports in the range are used, no port is available and the connection fails.

  Solution: Set **spark.port.maxRetries** to **50** and adjust the random port range of the executor to **spark.random.port.max** plus 100.

- View Spark configuration parameters:

  Run the **cat spark-env.sh** command on the client node to check whether the **SPARK_LOCAL_HOSTNAME** value is the IP address of the local host.

  This problem may occur when the client is directly copied from another node and the configuration parameters are not modified.

  Change the value of **SPARK_LOCAL_HOSTNAME** to the IP address of the local host.

  Note: If the cluster uses EIPs for communication, you need to add the following configuration:

  a. Add **spark.driver.host=***Elastic IP address of the client node* to **spark-default.conf**.

  b. Add **spark.driver.bindAddress=***IP address of the local host* to **spark-default.conf**.

c. Add **SPARK_LOCAL_HOSTNAME=***Elastic IP address of the client node* to **spark-env.sh**.

- If the communication and configuration are normal, check the code.

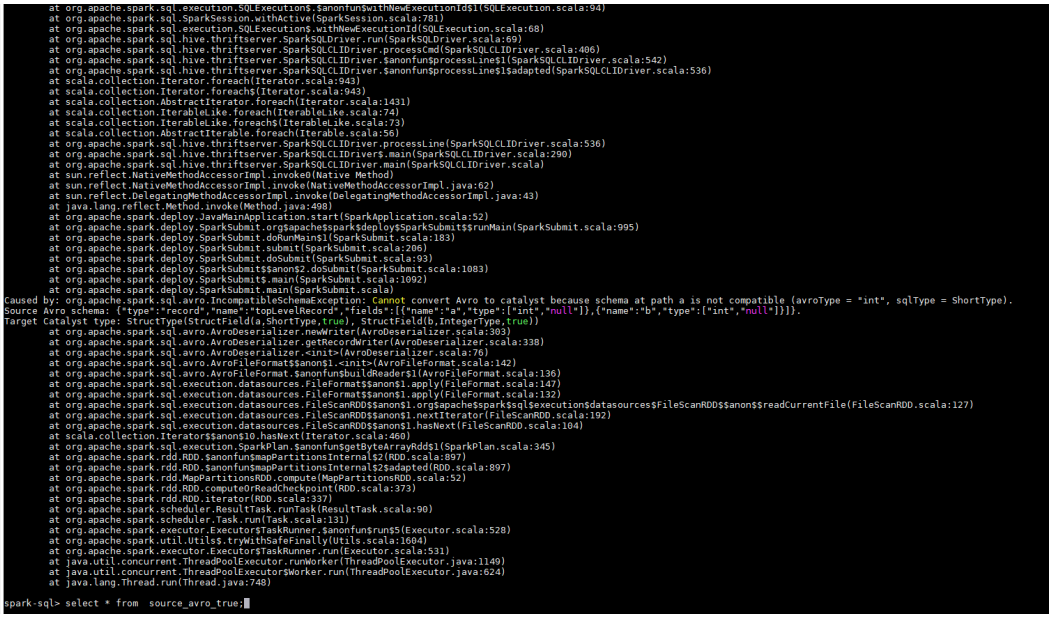
  When Spark starts a task, sparkDriverEnv is created on the client and bound to DRIVER_BIND_ADDRESS. This logic does not go to the server. So, this problem occurs because sparkDriver cannot obtain the corresponding host IP address due to abnormal OS environment of the client node.

  You can run the **export SPARK_LOCAL_HOSTNAME=172.0.0.1** command or set **spark.driver.bindAddress** to **127.0.0.1** so that the driver that submits tasks can load loopbackAddress.

24.17.15 How Do I Handle the Exception Occurred When I Query Datasource Avro Formats?

Question

An error is reported when I query Datasource Avro formats, and the message "Caused by: org.apache.spark.sql.avro.IncompatibleSchemaException" is displayed.



Answer

The datasource Avro formats are not compatible with the current data formats.

1. For new Avro files, set **spark.sql.forceConvertSchema.enabled** to **true** before you create them. This forcibly converts Avro formats to the specified data types and changes the schema at a time.
2. For existing Avro files, set **spark.sql.forceConvertSchema.enabled** to **true** before the query. If the query fails, run the **refresh table** command to clear the cache and then set query parameters. The Avro formats are forcibly converted to the specified data types, and the schema is temporarily modified on the client.

24.17.16 What Should I Do If Statistics of Hudi or Hive Tables Created Using Spark SQLs Are Empty Before Data Is Inserted?

Question

When Spark SQLs are used to create Hudi or Hive tables, the table statistics are empty before data is inserted.

Answer

You can use either of the following methods to collect the statistics:

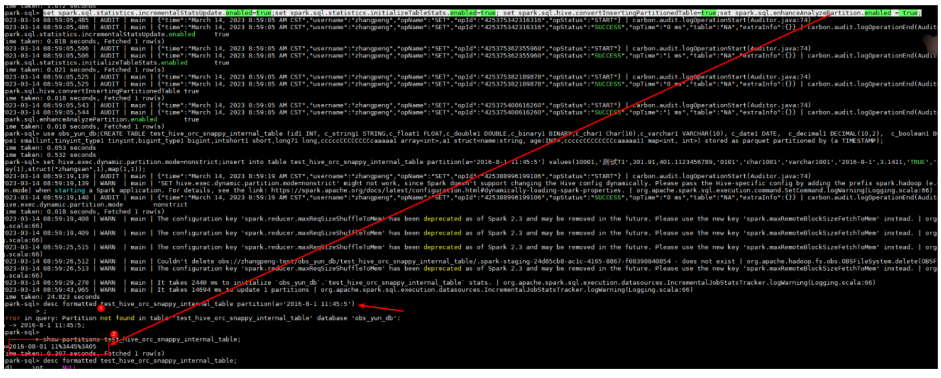
1. Run the **analyze** command to trigger statistics collection. If no data is inserted, run the **desc formatted table_name** command to check whether the value of **totalsize** is **0** after the **analyze** command is executed.
2. Set **spark.sql.statistics.size.autoUpdate.enabled** to **true** and insert data. Statistics collection will be triggered in the background.

24.17.17 Failed to Query Table Statistics by Partition Using Non-Standard Time Format When the Partition Column in the Table Creation Statement is timestamp

Question

When the partition column in the table creation statement is timestamp, the table statistics failed to be queried by partition using non-standard time format, and the result code of **show partitions table** is incorrect.

Run the **desc formatted test_hive_orc_snappy_internal_table partition(a='*2016-8-1 11:45:5*');** command to query the error. The following figure shows as an example.



Answer

The **spark.sql.hive.convertInsertingPartitionedTable** switch controls the insert and writing logic of Hive and Datasource tables. When Hive tables are used, timestamps are not automatically formatted. When Datasource tables are used, timestamps are automatically formatted.

If the written partition field is **a='2016-8-1 11:45:5'**, an error is reported, and it is automatically formatted to **a='2016-08-01 11:45:05'**.

To correctly query the table statistics, perform the following operation:

If the value of **spark.sql.hive.convertInsertingPartitionedTable** is set to **true**, use the data source table logic. You can run the following command to query the statistics:

desc formatted test_hive_orc_snappy_internal_table partition(a='*2016-08-01 11:45:05***');**

24.17.18 How Do I Use Special Characters with TIMESTAMP and DATE?

Question

In versions later than Spark 3.2.0, **TIMESTAMP(*)** or **DATE(*)** is not supported. The asterisk (*) can be any of the following characters:

- epoch
- today
- yesterday
- tomorrow
- now

The default format supported is either the timestamp '*' or data '*' format. If the previous syntax is used to insert data into the data table, a NULL value will be returned.

Answer

To make the **spark.sql.convert.special.datetime** parameter compatible with the previous syntax, run the following command on the Spark client:

set spark.sql.convert.special.datetime=true;

```
spark-sql> set spark.sql.convert.special.datetime=true;
spark.sql.convert.special.datetime      true
Time taken: 0.035 seconds, Fetched 1 row(s)
```


25 Using Sqoop

25.1 Using Sqoop from Scratch

Sqoop is an open-source tool that transfers data between Hadoop (Hive) and traditional databases like MySQL and PostgreSQL. It can import data from relational databases, including MySQL, Oracle, and PostgreSQL, to Hadoop HDFS. It can also import HDFS data to a relational database.

Prerequisite

- The Sqoop component and dependent services have been selected during cluster creation.
- Install the client. For details, see **Installing a Client (MRS 3.x or Later)**. To facilitate the use of Sqoop, you need to install a full client when using Sqoop. If the installation directory is **/opt/client**, change it to the actual installation directory.
- If Kerberos authentication has been enabled for the cluster, you need to create or obtain a user with the required permission to run Sqoop commands. The created user must be added to the **hadoop**, **supergroup**, and **hive** groups. For details, see **Creating a User**.
- The JDBC driver package (for example, **mysql-connector-java-5.1.47.jar**) for the corresponding database has been saved in the client directory **/Sqoop/sqoop/lib**. The permission and user group have been adjusted to match other JAR packages in the directory.

Exporting Data from HDFS to MySQL

Step 1 Log in to the node where the Sqoop client is deployed.

Step 2 Initialize environment variables.

source /opt/client/bigdata_env

Step 3 If Kerberos authentication is enabled for the cluster, run the **kinit** command to authenticate the user. If Kerberos authentication is not enabled for the cluster, go to **Step 4**.

For example, the **sqoop_user** user is authenticated using the following command:

kinit sqoop_user

Step 4 Operate the Sqoop client.

sqoop export --connect jdbc:mysql://10.100.*xxx.xxx***:***3306/test* **--username root --password** *xxx* **--table** *component13* **-export-dir hdfs://hacluster/***user/hive/warehouse/component_test3* **--fields-terminated-by ',' -m 1**

For details about more parameters, see **Common Sqoop Commands and Parameters**.

Table 25-1 Parameters

Parameter	Description
--connect	Specifies the URL for connecting to JDBC. The value format is **jdbc:mysql://***IP address of the MySQL database*:*MySQL Port*/*Database name*.
--username	Specifies the username for connecting to the MySQL database.
-password	Specifies the password for connecting to the MySQL database. To prevent any potential information leaks that may arise from commands containing authentication passwords, it is advisable to disable the command history recording function before executing any commands to avoid security risks.
-table <table-name>	Specifies the name of the MySQL table used to store exported data.
-export-dir <dir>	Specifies the HDFS path of the Sqoop table to be exported.
--fields-terminated-by	Specifies the delimiter of the exported data, which must be the same as that in the HDFS data table to be exported.
-m or -num-mappers <n>	Starts *n* (4 by default) maps to import data concurrently. The value cannot be greater than the maximum number of maps in a cluster.
-direct	Imports data to a relational database using a database import tool, for example, mysqlimport of MySQL, more efficient than the JDBC connection mode.
-update-key <col-name>	Specifies the column used for updating the existing data in a relational database.
-update-mode <mode>	Specifies how updates are performed. The value can be **updateonly** or **allowinsert**. This parameter is used only when the relational data table does not contain the data record to be imported. For example, if the HDFS data to be imported to the destination table contains a data record **id=1** and the table contains an existing data record **id=2**, the update will fail.

Parameter	Description
-input-null-string <null-string>	This parameter is optional. If it is not specified, **null** will be used.
-input-null-non-string <null-string>	This parameter is optional. If it is not specified, **null** will be used.
-staging-table <staging-table-name>	Creates a table with the same data structure as the destination table for storing data before it is imported to the destination table. This parameter is used to ensure transaction security during data import to relational database tables. If multiple transactions exist during data import, the failure of one transaction can impact others. This parameter can be used to prevent issues caused by incorrect or duplicate data records.
-clear-staging-table	Clears data in the staging table before data is imported if the staging-table is not empty.

----End

Importing Data from MySQL to Hive

Step 1 Log in to the node where the Sqoop client is deployed.

Step 2 Initialize environment variables.

source /opt/client/bigdata_env

Step 3 If Kerberos authentication is enabled for the cluster, run the **kinit** command to authenticate the user. If Kerberos authentication is not enabled for the cluster, go to **Step 4**.

For example, the **sqoop_user** user is authenticated using the following command:

kinit sqoop_user

Step 4 Operate the Sqoop client.

sqoop import --connect jdbc:mysql://10.100.*xxx.xxx***:3306/***test* **--username root --password** *xxx* **--table component --hive-import --hive-table** *component_test2* **--delete-target-dir --fields-terminated-by "," -m 1 --as-textfile**

Table 25-2 Parameters

Parameter	Description
--hive-import	Imports data from a relational database to MRS Hive.
--delete-target-dir	Deletes the existing target file (if any) from Hive and imports again.

Parameter	Description
-append	Appends data to an existing dataset in HDFS. Once this parameter is used, Sqoop imports data to a temporary directory, renames the temporary file where the data is stored, and moves the file to a formal directory to avoid duplicate file names in the directory.
-as-avrodatafile	Imports data to a data file in the Avro format.
-as-sequencefile	Imports data to a sequence file.
-as-textfile	Imports data to a text file. After the text file is generated, you can run SQL statements in Hive to query the result.
--as-parquetfile	Imports data to a Parquet file.
-boundary-query <statement>	Specifies the SQL statement for performing boundary query. Before importing data, use a SQL statement to obtain a result set and import the data in the result set. The data format can be **-boundary-query 'select id,creationdate from person where id = 3'** (indicating a data record whose ID is 3) or **select min(<split-by>), max(<split-by>) from <table name>**. The fields to be queried cannot contain fields whose data type is string. Otherwise, the error message "java.sql.SQLException: Invalid value for getLong()" is displayed.
-columns<col,col,col...>	Specifies the fields to be imported. The format is ***-Column id,Username***.
-direct	Imports data to a relational database using a database import tool, for example, mysqlimport of MySQL, more efficient than the JDBC connection mode.
-direct-split-size	Splits the imported streams by byte. Especially when data is imported from PostgreSQL using the direct mode, a file that reaches the specified size can be divided into several independent files.
-inline-lob-limit	Sets the maximum value of an inline LOB.
-m or -num-mappers	Starts *n* (4 by default) maps to import data concurrently. The value cannot be greater than the maximum number of maps in a cluster.

Parameter	Description
-query, -e<statement>	Imports data from the query result. To use this parameter, you must specify the **-target-dir** and **-hive-table** parameters and use the query statement containing the **WHERE** clause as well as **$CONDITIONS**. Example: **-query'select * from person where $CONDITIONS' -target-dir /user/hive/warehouse/person -hive-table person**
-split-by<column-name>	Specifies the column of a table used to split work units. Generally, the column name is followed by the primary key ID.
-table <table-name>	Specifies the relational database table from which data is obtained.
-target-dir <dir>	Specifies the HDFS path.
-warehouse-dir <dir>	Specifies the directory for storing data to be imported. This parameter is applicable when data is imported to HDFS but cannot be used when you import data to Hive directories. This parameter cannot be used together with **-target-dir**.
-where	Specifies the **WHERE** clause when data is imported from a relational database, for example, **-where 'id = 2'**.
-z,-compress	Compresses sequence, text, and Avro data files using the GZIP compression algorithm. Data is not compressed by default.
–compression-codec	Specifies the Hadoop compression codec. GZIP is used by default.
–null-string <null-string>	Specifies the string to be interpreted as **NULL** for string columns.
–null-non-string<null-string>	Specifies the string to be interpreted as null for non-string columns. If this parameter is not specified, **NULL** will be used.
-check-column (col)	Specifies the column for checking incremental data import, for example, **id**.
-incremental (mode) append or lastmodified	Incrementally imports data. **append**: appends records, for example, appending records that are greater than the value specified by **last-value**. **lastmodified**: appends data that is modified after the date specified by **last-value**.

Parameter	Description
-last-value (value)	Specifies the maximum value (greater than the specified value) of the column after the last import. This parameter can be set as required.

----End

Sqoop Usage Example

- Importing data from MySQL to HDFS using the **sqoop import** command

  sqoop import --connect jdbc:mysql://10.100.231.134:3306/test --username root --password *xxx* **--query 'SELECT * FROM component where $CONDITIONS and component_id ="MRS 1.0_002"' --target-dir /tmp/component_test --delete-target-dir --fields-terminated-by "," -m 1 --as-textfile**

- Exporting data from OBS to MySQL using the **sqoop export** command

  sqoop export --connect jdbc:mysql://10.100.231.134:3306/test --username root --password *xxx* **--table component14 -export-dir obs://obs-file-bucket/xx/part-m-00000 --fields-terminated-by ',' -m 1**

- Importing data from MySQL to OBS using the **sqoop import** command

  sqoop import --connect jdbc:mysql://10.100.231.134:3306/test --username root --password *xxx* **--table component --target-dir obs://obs-file-bucket/xx --delete-target-dir --fields-terminated-by "," -m 1 --as-textfile**

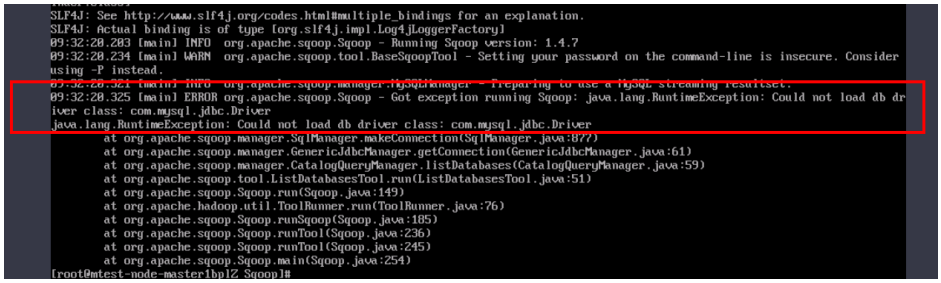
- Importing data from MySQL to OBS tables outside Hive using the **sqoop import** command

  sqoop import --connect jdbc:mysql://10.100.231.134:3306/test --username root --password *xxx* **--table component --hive-import --hive-table component_test01 --fields-terminated-by "," -m 1 --as-textfile**

MySQL Driver Package Is Missing During Data Import or Export

If the error "Could not load db driver class: com.mysql.jdbc.Driver" is reported when you run the **sqoop import** or **sqoop export** command, the MySQL driver package is missing. Download the MySQL driver package from the MySQL official website, decompress it, upload it to the *Client installation directory*/**Sqoop/sqoop/lib**, and run the command again.

Figure 25-1 Error indicating that the MySQL driver package is missing



25.2 Common Sqoop Commands and Parameters

For more Sqoop command parameters, see the Sqoop official document at **https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html**.

Common Commands

Table 25-3 Common commands

Command	Description
import	Imports data to a cluster.
export	Exports data of a cluster.
codegen	Obtains data from a table in the database to generate a Java file and compress the file.
create-hive-table	Creates a Hive table.
eval	Executes a SQL statement and view the result.
import-all-tables	Imports all tables in a database to HDFS.
job	Generates a Sqoop job.
list-databases	Lists database names.
list-tables	List table names.
merge	Merges data in different HDFS directories and saves the data to a specified directory.
metastore	Starts the metadata database to record the metadata of a Sqoop job.
help	Prints help information.
version	Prints the version information.

Common Parameters

Table 25-4 Common parameters

Type	Parameter	Description
Database connection	--connect	Specifies the URL for connecting to a relational database.
	--connection-manager	Specifies the connection manager class.

Type	Parameter	Description
	--driver jdbc	Specifies the driver package for database connection.
	--help	Prints help information.
	--password	Specifies the password for connecting to a database.
	--username	Specifies the username for connecting to a database.
	--verbose	Prints detailed information on the console.
import parameters	--fields-terminated-by	Specifies the field delimiter, which must be the same as that in a Hive table or HDFS file.
	--lines-terminated-by	Specifies the line delimiter, which must be the same as that in a Hive table or HDFS file.
	--mysql-delimiters	Specifies the default delimiter settings of MySQL.
export parameters	--input-fields-terminated-by	Specifies the field delimiter.
	--input-lines-terminated-by	Specifies the line delimiter.
Hive parameters	--hive-delims-replacement	Replaces characters such as **\r** and **\n** in data with user-defined characters.
	--hive-drop-import-delims	Removes characters such as **\r** and **\n** when data is imported to Hive.
	--map-column-hive	Specifies the data type of fields during the generation of a Hive table.
	--hive-partition-key	Creates a partition.
	--hive-partition-value	Imports data to a specified partition of a database.
	--hive-home	Specifies the installation directory for Hive.
	--hive-import	Specifies that data is imported from a relational database to Hive.
	--hive-overwrite	Overwrites existing Hive data.

Type	Parameter	Description
	--create-hive-table	Creates a Hive table. The default value is **false**. A destination table will be created if it does not exist.
	--hive-table	Specifies a Hive table to which data is to be imported.
	--table	Specifies the relational database table.
	--columns	Specifies the fields of a relational data table to be imported.
	--query	Specifies the query statement for importing the query result.
HCatalog parameters	--hcatalog-database	Specifies a Hive database and imports data to it using HCatalog.
	--hcatalog-table	Specifies a Hive table and imports data to it using HCatalog.
Others	-m or --num-mappers	Specifies the number of map tasks used by a Sqoop job.
	--split-by	Specifies the column based on which Sqoop splits work units. This parameter is used together with **-m**.
	--target-dir	Specifies the temporary directory of HDFS.
	--null-string string	Specifies the string to be written for a null value for string columns.
	--null-non-string	Specifies the string to be written for a null value for non-string columns.
	--check-column	Specifies the column for determining incremental data import.
	--incremental append or lastmodified	Incrementally imports data. **append**: appends records, for example, appending records that are greater than the value specified by **last-value**. **lastmodified**: appends data that is modified after the date specified by **last-value**.
	--last-value	Specifies the last value of the check column from the previous import.
	--input-null-string	Specifies the string to be interpreted as **NULL** for string columns.

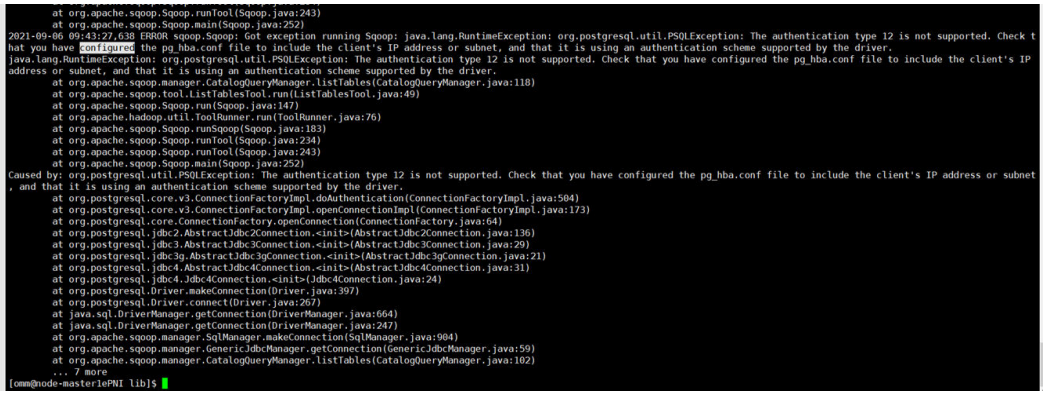
Type	Parameter	Description
	--input-null-non-string	Specifies the string to be interpreted as null for non-string columns. If this parameter is not specified, **NULL** will be used.

25.3 Sqoop FAQs

25.3.1 What Should I Do If PostgreSQL or GaussDB Failed to Be Connected?

Question

An error is reported when PostgreSQL or GaussDB is connected.



Answer

Scenario 1: (**import** scenarios) Run the **sqoop import** command to extract the open source PostgreSQL to MRS HDFS or Hive.

- Symptom:

  The **sqoop** command can be executed to query PostgreSQL tables, but an error is reported when the **sqoop import** command is executed.

  ```
  The authentication type 5 is not supported. Check that you have configured the pg_hba.conf file to include the client's IP address or subnet, and that it
  The authentication type 12 is not supported. Check that you have configured the pg_hba.conf file to include the client's IP address or subnet, and that it
  ```

- Root cause:
  - If the authentication type is 5, the root cause is as follows: When the **sqoop import** command is executed, a MapReduce job is started. The PostgreSQL driver package **gsjdbc4-*.jar** exists in the MRS Hadoop installation directory **${BIGDATA_HOME}/FusionInsight_HD_*/1_*_NodeManager/install/hadoop/share/hadoop/common/lib**, which is incompatible with the open source PostgreSQL service. As a result, an error is reported.

  - If the authentication type is 12, the root cause is as follows: The **pg_hba.conf** file of the database is incorrectly configured.

- Answer:
  - If the authentication type is 5, the solution is as follows: Move the driver package **gsjdbc4-*.jar** to the **tmp** directory on each node where MRS NodeManager instance is deployed.

    mv ${BIGDATA_HOME}/FusionInsight_HD_*/1_*_NodeManager/install/hadoop/share/hadoop/common/lib/gsjdbc4-*.jar /tmp
  - If the authentication type is 12, the solution is as follows: Modify the **pg_hba.conf** file of the database by changing the value of **ADDRESS** to the IP address of the node where Sqoop resides.

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all             all             0.0.0.0/0               md5
# IPv6 local connections:
host    all             all             ::1/128                 trust
#host    all             all             0.0.0.0/0               password
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     postgres                                trust
host    replication     postgres        127.0.0.1/32            trust
host    replication     postgres        ::1/128                 trust
```

Scenario 2: (**export** scenarios) Run the **sqoop export** command to extract the open source PostgreSQL to MRS HDFS or Hive.

- Symptom:

  The **sqoop** command can be executed to query PostgreSQL tables, but an error is reported when the **sqoop export** command is executed.

  ```
  The authentication type 5 is not supported. Check that you have configured the pg_hba.conf file to include the client's IP address or subnet, and that it
  ```

- Root cause:

  When the **sqoop export** command is executed, a MapReduce job is started. The PostgreSQL driver package **gsjdbc4-*.jar** exists in the MRS Hadoop installation directory **${BIGDATA_HOME}/FusionInsight_HD_*/1_*_NodeManager/install/hadoop/share/hadoop/common/lib**, which is incompatible with the open-source PostgreSQL service. As a result, an error is reported.

- Answer:

  a. Move the driver package **gsjdbc4-*.jar** to the **tmp** directory on each node where MRS NodeManager instance is deployed.

  mv ${BIGDATA_HOME}/FusionInsight_HD_*/1_*_NodeManager/install/hadoop/share/hadoop/common/lib/gsjdbc4-*.jar /tmp

b. Delete **/opt/client/Hive/Beeline/lib/gsjdbc4-*.jar**.

25.3.2 What Should I Do If Data Failed to Be Synchronized Using hive-table?

Question

An error is reported when data is synchronized in hive-table mode.

```
        at org.apache.hadoop.hive.ql.metadata.Hive.registerAllFunctionsOnce(Hive.java:260)  [hive-exec-3.1.0-hw-ei-312001-SNAPSHOT.jar:3.1.0-hw-ei-312001-SNAPSHOT]
        ... 41 more
14:41:42.891 [bfe1438c-07bb-43fd-91d9-910a348f6e91 main] ERROR org.apache.hadoop.hive.metastore.ObjectStore - Version information not found in metastore. The process will exit.
14:41:42.892 [bfe1438c-07bb-43fd-91d9-910a348f6e91 main] ERROR org.apache.hadoop.hive.metastore.RetryingHMSHandler - ExitSecurityException
        at org.apache.sqoop.util.SubprocessSecurityManager.checkExit(SubprocessSecurityManager.java:83)
        at java.lang.Runtime.exit(Runtime.java:107)
        at java.lang.System.exit(System.java:973)
        at org.apache.hadoop.hive.metastore.ObjectStore.checkSchema(ObjectStore.java:9555)
        at org.apache.hadoop.hive.metastore.ObjectStore.verifySchema(ObjectStore.java:9531)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.hive.metastore.RawStoreProxy.invoke(RawStoreProxy.java:97)
        at com.sun.proxy.$Proxy37.verifySchema(Unknown Source)
        at org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.getMSForConf(HiveMetaStore.java:903)
        at org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.getMS(HiveMetaStore.java:895)
        at org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.createDefaultDB(HiveMetaStore.java:978)
        at org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.init(HiveMetaStore.java:585)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.hive.metastore.RetryingHMSHandler.invokeInternal(RetryingHMSHandler.java:148)
        at org.apache.hadoop.hive.metastore.RetryingHMSHandler.invoke(RetryingHMSHandler.java:109)
        at org.apache.hadoop.hive.metastore.RetryingHMSHandler.<init>(RetryingHMSHandler.java:81)
        at org.apache.hadoop.hive.metastore.RetryingHMSHandler.getProxy(RetryingHMSHandler.java:94)
        at org.apache.hadoop.hive.metastore.HiveMetaStore.newRetryingHMSHandler(HiveMetaStore.java:9683)
        at org.apache.hadoop.hive.metastore.HiveMetaStoreClient.<init>(HiveMetaStoreClient.java:185)
        at org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClient.<init>(SessionHiveMetaStoreClient.java:96)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
        at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
        at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
        at org.apache.hadoop.hive.metastore.utils.JavaUtils.newInstance(JavaUtils.java:84)
        at org.apache.hadoop.hive.metastore.RetryingMetaStoreClient.<init>(RetryingMetaStoreClient.java:97)
```

Answer

Add the following content to the **hive-site.xml** file.

```
</property>
<property>
<name>hive.metastore.schema.verification</name>
<value>false</value>
</property>
```

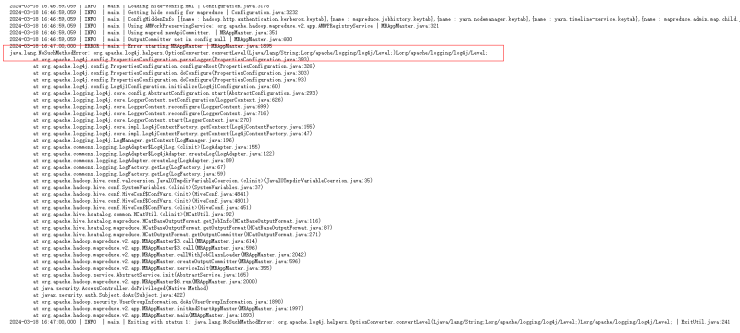
25.3.3 What Should I Do If An Error Is Reported When Data Is Imported to a Hive Table?

Question

The following error log is printed when Sqoop imports data to the Hive table.


```
2024-03-18 16:47:07,861 INFO mapreduce.Job: Job job_1710577134946_0080 running in uber mode : false
2024-03-18 16:47:07,862 INFO mapreduce.Job:  map 0% reduce 0%
2024-03-18 16:47:07,873 INFO mapreduce.Job: Job job_1710577134946_0080 failed with state FAILED due to: Application application_1710577134946_00
80 failed 2 times (global limit =5; local limit is =2) due to AM Container for appattempt_1710577134946_0080_000002 exited with  exitCode: 1
Failing this attempt.Diagnostics: [2024-03-18 16:47:07.784]Exception from container-launch.
Container id: container_1710577134946_0080_02_000001
Exit code: 1
Exception message: Launch container failed
Shell error output: Nonzero exit code=1, error message='Invalid argument number'

Shell output: main : command provided 1
main : run as user is super
main : requested yarn user is super
Getting exit code file...
Creating script paths...
Writing pid file...
Writing to tmp file /srv/BigData/data1/nm/localdir/nmPrivate/application_1710577134946_0080/container_1710577134946_0080_02_000001/container_171
0577134946_0080_02_000001.pid.tmp
Writing to cgroup task files...
Creating local dirs...
Launching container...


[2024-03-18 16:47:07.786]Container exited with a non-zero exit code 1. Error file: prelaunch.err.
Last 4096 bytes of prelaunch.err :
Last 4096 bytes of stderr :
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/Bigdata/FusionInsight_HD_8.3.1/install/FusionInsight-Hadoop-3.3.1/hadoop/share/hadoop/common/lib/slf4j-re
load4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/srv/BigData/data1/nm/localdir/usercache/super/filecache/26/libjars/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/imp
l/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/srv/BigData/data1/nm/localdir/usercache/super/filecache/26/libjars/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/
StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Reload4jLoggerFactory]
log4j:WARN No appenders could be found for logger (org.apache.hadoop.mapreduce.v2.app.MRAppMaster).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

Check the appattempt logs for the Yarn application. The syslog includes task logs as follows:



Answer

The log framework of Hive conflicts with that of Hadoop. Delete **log4j-*-api-*.jar** from the Hive directory in the current Sqoop client directory.

1. Log in to the client node. For example, if the client directory is **/opt/client**, run the following command:

   rm -rf /opt/client/Hive/Beeline/lib/log4j-*-api-*.jar

2. Run the command again.

26 Using Tez

26.1 Accessing the Tez Web UI to View the Task Execution Result

Tez web UI displays the Tez task execution process on a GUI. You can view the task execution details on the GUI.

Prerequisite

The TimelineServer instance of the YARN service has been installed in the current MRS cluster.

Log in to the Tez web UI.

Log in to Manager. For details, see **Accessing FusionInsight Manager**. On Manager, choose **Cluster** > **Services** > **Tez**. Click the link on the right of **Tez WebUI** in the **Basic Information** area, and go to Tez web UI. You can view the details about Tez task execution.

26.2 Typical Tez Configuration Parameters

Navigation path for setting parameters:

On Manager, choose **Cluster** > **Service** > **Tez** > **Configuration** > **All Configurations**.

Enter a parameter name in the search box.

Parameter description

Table 26-1 Parameter description

Parameter	Description	Default Value
property.tez.log.dir	TezUI log directory	/var/log/Bigdata/tez/tezui
property.tez.log.level	TezUI log level	INFO

26.3 Tez Log Overview

Log Description

Log path: The default save path of Tez logs is **/var/log/Bigdata/tez/***role name*.

TezUI: **/var/log/Bigdata/tez/tezui** (run logs) and **/var/log/Bigdata/audit/tez/tezui** (audit logs)

Log archive rule: The automatic compression and archiving function of Tez is enabled. By default, when the size of a log file exceeds 20 MB (which is adjustable), the log file is automatically compressed. The naming rule of the compressed log file is as follows: <*Original log file name*>-<*yyyy-mm-dd_hh-mm-ss*>.[*ID*].**log.zip** A maximum of 20 latest compressed files are retained. The number of compressed files and compression threshold can be configured.

Table 26-2 Tez log list

Log Type	Name	Description
Run log	tezui.out	Log file that records TezUI running environment information
	tezui.log	Run log of the TezUI process
	tezui-omm-<*Date*>-gc.log.<*No.*>	GC log of the TezUI process
	prestartDetail.log	Work logs generated before the TezUI is started
	check-serviceDetail.log	Log file that records whether the TezUI service starts successfully
	postinstallDetail.log	Work logs after the TezUI is installed
	startDetail.log	Startup log of the TezUI process

Log Type	Name	Description
	stopDetail.log	Stop log of the TezUI process
Audit log	tezui-audit.log	TezUI audit log

Log Level

Table 26-3 describes the log levels supported by TezUI.

Levels of run logs are ERROR, WARN, INFO, and DEBUG from the highest to the lowest priority. Run logs of equal or higher levels are recorded. The higher the specified log level, the fewer the logs recorded.

Table 26-3 Log levels

Level	Description
ERROR	Logs of this level record error information about system running.
WARN	Exception information about the current event processing
INFO	Logs of this level record normal running status information about the system and events.
DEBUG	Logs of this level record the system information and system debugging information.

To modify log levels, perform the following operations:

Step 1 Log in to Manager.

Step 2 Choose **Cluster** > **Service** > **Tez** > **Configuration**.

Step 3 Select **All Configurations**.

Step 4 In the navigation pane, choose **TezUI** > **Log**.

Step 5 Select a desired log level.

Step 6 Click **Save**. In the dialog box that is displayed, click **OK** to save the configuration.

Step 7 Click **Instance**, select the **TezUI** role, choose **More** > **Restart Instance**, enter the user password, and click **OK** in the dialog box that is displayed.

Step 8 Wait until the instance is restarted for the configuration to take effect.

----End

Log Format

The following table lists the Tez log formats.

Table 26-4 Log formats

Log Type	Format	Example
Run log	<yyyy-MM-dd HH:mm:ss,SSS>|<LogLevel>|<Thread that generates the log>|<Message in the log>|<Location of the log event>	2020-07-31 11:44:21,378 | INFO | TezUI-health-check | Start health check | com.XXX.tez.HealthCheck.run(HealthCheck.java:30)
Audit logs	<yyyy-MM-dd HH:mm:ss,SSS>|<LogLevel>|<Thread that generates the log>|<User Name><User IP><Time><Operation><Resource><Result><Detail >|<Location of the log event >	2018-12-24 12:16:25,319 | INFO | HiveServer2-Handler-Pool: Thread-185 | UserName=hive UserIP=10.153.2.204 Time=2018/12/24 12:16:25 Operation=CloseSession Result=SUCCESS Detail= | org.apache.hive.service.cli.thrift.ThriftCLIService.logAuditEvent(ThriftCLIService.java:434)

26.4 Common Issues About Tez

26.4.1 Tez Task Details Cannot Be Displayed on the Tez Web UI

Question

After a user logs in to Manager and switches to the Tez web UI, the submitted Tez tasks are not displayed.

Answer

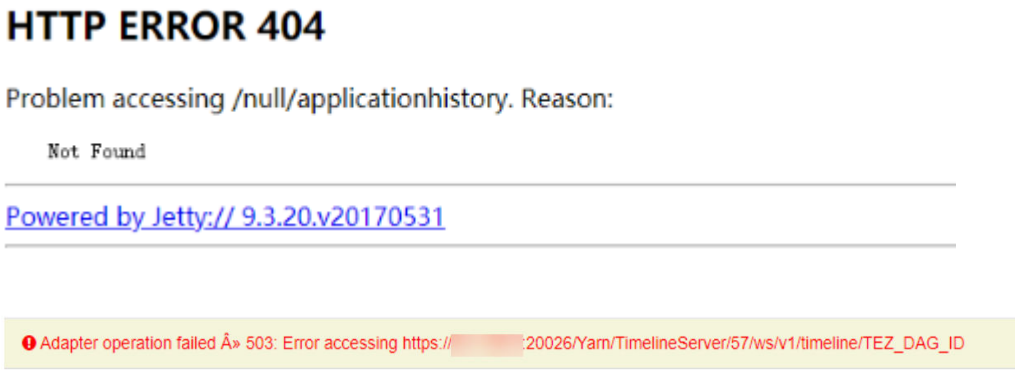
The Tez task data displayed on the Tez WebUI requires the support of TimelineServer of YARN. Ensure that TimelineServer has been enabled and is running properly before the task is submitted.

When setting the Hive execution engine to Tez, you need to set **yarn.timeline-service.enabled** to **true**. For details, see **Switching the Hive Execution Engine to Tez**.

26.4.2 Failed to Access the Tez Web UI

Question

When a user logs in to Manager and switches to the Tez web UI, error 404 or 503 is displayed.



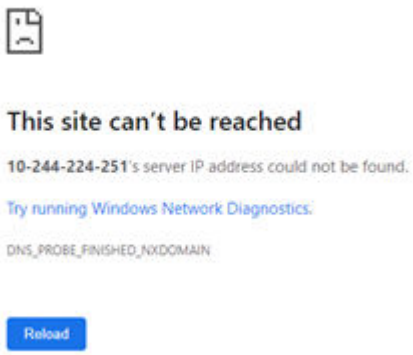
Answer

The Tez web UI depends on the TimelineServer instance of YARN. Therefore, TimelineServer must be installed in advance and in the **Good** state.

26.4.3 YARN Logs Cannot Be Viewed on the Tez Web UI

Question

A user logs in to the Tez web UI and clicks **Logs**, but the YARN log page fails to be displayed and data cannot be loaded.



Answer

Currently, the hostname is used for the access to the YARN log page from the Tez web UI. Therefore, you need to configure the mapping between the hostname and IP address on the Windows host.

Add the mapping between the host name and the IP address to the **hosts** file located in the **C:\Windows\System32\drivers\etc** directory on the Windows host. Save the file and access the host again.

The following is an example:

```
10.244.224.45 10-044-224-45
```

26.4.4 Table Data Is Empty on the TezUI HiveQueries Page

Question

A user logs in to Manager and switches to the Tez web UI page, but no data for the submitted task is displayed on the **Hive Queries** page.

Answer

To display task data on the **Hive Queries** page on the Tez web UI, you need to set the following parameters:

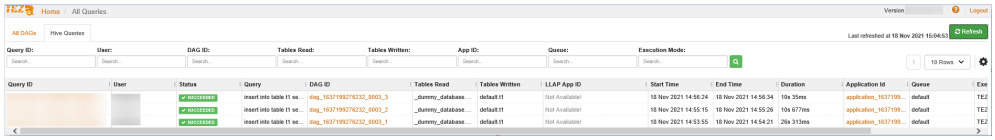
On FusionInsight Manager, choose **Cluster** > **Service** > **Hive** and click the **Configurations** tab and then **All Configurations**. In the navigation pane on the left, choose **HiveServer** > **Customization**. Add the following configuration to **hive-site.xml**:

Attribute	Attribute Value
hive.exec.pre.hooks	org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.post.hooks	org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.failure.hooks	org.apache.hadoop.hive.ql.hooks.ATSHook

NOTE

Data display on TezUI depends on the TimelineServer instance of Yarn. If the TimelineServer instance is faulty or not started, you need to set **yarn.timeline-service.enabled** to **false** in **yarn-site.xml**. Otherwise, the Hive task fails to be executed.

After you configure the parameters and re-execute the Hive task, data can be displayed on the **Hive Queries** page. However, data of previous tasks cannot be displayed.



27 Using YARN

27.1 Yarn User Permission Management

27.1.1 Creating Yarn Roles

Scenario

Create and configure a YARN role. The Yarn role can be assigned with Yarn administrator permission and manage Yarn queue resources.

NOTE

If the current component uses Ranger for permission control, you need to configure permission management policies based on Ranger. Refer to **Adding a Ranger Access Permission Policy for Yarn**.

Prerequisites

- The MRS cluster administrator has understood service requirements.
- You have logged in to Manager.

Procedure

Step 1 Choose System > Permission > Role.

Step 2 Click **Create Role** and set a role name and enter description.

Step 3 Refer **Table 27-1** to configure resource permissions for roles.

Yarn permissions:

- Cluster management: Yarn administrator permissions.
- Queue scheduling: queue resource management.

Table 27-1 Setting a role

Task	Operation
Setting the Yarn administrator permission	In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Yarn** > **Cluster Management**. **NOTE** The Yarn service needs to be restarted to set the Yarn administrator permission so that the saved role configuration can take effect.
Setting the permission for a user to submit tasks in a specified Yarn queue	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Yarn** > **Scheduling Queue** > **root**. 2. In the **Permission** column of the specified queue, select **Submit**.
Setting the permission for a user to manage tasks in a specified Yarn queue	1. In the **Configure Resource Permission** table, choose *Name of the desired cluster* > **Yarn** > **Scheduling Queue** > **root**. 2. In the **Permission** column of the specified queue, select **Manage**.

If the Yarn role contains the **Submit** or **Manage** permission of a parent queue, the sub-queue inherits the permission by default, that is, the **Submit** or **Manage** permission is automatically added for the sub-queue. Permissions inherited by sub-queues will not be displayed as selected in the **Configure Resource Permission** table.

If you select only the **Submit** permission of a parent queue when setting the Yarn role, you need to manually specify the queue name when submitting tasks as a user with the permission of this role. Otherwise, when the parent queue has multiple sub-queues, the system does not automatically determine the queue to which the task is submitted and therefore submits the task to the **default** queue.

Step 4 Click **OK**.

----End

27.2 Submitting a Task Using the Yarn Client

Scenario

This section guides users to use a Yarn client in an O&M or service scenario.

Prerequisites

- The client has been installed.

  For example, the installation directory is **/opt/client**. The client directory in the following operations is only an example. Change it based on the actual installation directory onsite.

- Service component users have been created by the MRS cluster administrator. In security mode, machine-machine users need to download the keytab file. A human-machine user must change the password upon the first login. In common mode, you do not need to download the keytab file or change the password.

Using the Yarn Client

Step 1 Install a client. For details, see **Installing a Client**.

Step 2 Log in to the node where the client is installed as the client installation user.

Step 3 Run the following command to go to the client installation directory:

cd /opt/client

Step 4 Run the following command to configure environment variables:

source bigdata_env

Step 5 If the cluster is in security mode, run the following command to authenticate the user. In normal mode, user authentication is not required.

kinit *Component service user*

Step 6 Run the Yarn command. The following provides an example:

yarn application -list

----End

Client-related FAQs

1. What Do I Do When the Yarn Client Exits Abnormally and Error Message "java.lang.OutOfMemoryError" Is Displayed After the Yarn Client Command Is Run?

   This problem occurs because the memory required for running the Yarn client exceeds the upper limit (128 MB by default) set on the Yarn client. You can modify **CLIENT_GC_OPTS** in *<Client installation path>***/HDFS/component_env** to change the memory upper limit of the Yarn client. For example, if you want to set the maximum memory to 1 GB, run the following command:

   ```
   export CLIENT_GC_OPTS="-Xmx1G"
   ```

   After the modification, run the following command to make the modification take effect:

   source <*Client installation path*>**/bigdata_env**

2. How Can I Set the Log Level When the Yarn Client Is Running?

   By default, the logs generated during the running of the Yarn client are printed to the console. The default log level is INFO. To enable the DEBUG log level for fault locating, run the following command to export an environment variable:

   export YARN_ROOT_LOGGER=DEBUG,console

   Then run the Yarn Shell command to print DEBUG logs.

   If you want to print INFO logs again, run the following command:

export YARN_ROOT_LOGGER=INFO,console

27.3 Configuring Container Log Aggregation

Scenario

Yarn provides the container log aggregation function to collect logs generated by containers on each node to HDFS to release local disk space. You can collect logs in either of the following ways:

- After the application is complete, collect container logs to HDFS at a time.
- During application running, periodically collect log segments generated by containers and save them to HDFS.

Configuration Description

Navigation path for setting parameters:

Go to the **All Configurations** tab page of YARN, enter the parameters listed in **Table 27-2** in the search box, modify the parameters by referring to **Modifying Cluster Service Configuration Parameters**, and save the configuration. On the **Dashboard** tab page, choose **More** > **Synchronize Configuration**. After the synchronization is complete, restart the YARN service.

The **yarn.nodemanager.remote-app-log-dir-suffix** parameter must be configured on the Yarn client. The configurations on the ResourceManager, NodeManager, and JobHistory nodes must be the same as those on the Yarn client.

The periodic log collection function applies only to MapReduce applications, for which rolling output of log files must be configured. **Table 27-4** describes the configurations in the *client installation path*/**Yarn/config/mapred-site.xml** configuration file on the MapReduce client node.

Table 27-2 Parameter description

Parameter	Description	Default Value
yarn.log-aggregation-enable	Whether to enable container log aggregation ● If this parameter is set to **true**, logs are collected to the HDFS directory. ● If this parameter is set to **false**, the function is disabled, and logs are not collected to HDFS. After changing the parameter value, restart the Yarn service for the setting to take effect. **NOTE** ● The container logs that are generated before the parameter is set to **false** and the setting takes effect cannot be obtained from the web UI. ● If you need to view the logs generated before on the web UI, you are advised to set this parameter to **true**.	true
yarn.nodemanager.log-aggregation.roll-monitoring-interval-seconds	Interval for NodeManager to periodically collect logs ● If this parameter is set to **-1** or **0**, periodic log collection is disabled. Logs are collected at a time after application running is complete. ● The minimum collection interval can be set to 3,600 seconds. If this parameter is set to a value greater than 0 and less than 3,600, the collection interval is 3,600 seconds. Interval for NodeManager to wake up and upload logs. If this parameter is set to **-1** or **0**, rolling monitoring is disabled and logs are aggregated when the application task is complete. The value must be greater than or equal to **-1**.	-1

Parameter	Description	Default Value
yarn.nodemanager.disk-health-checker.log-dirs.max-disk-utilization-per-disk-percentage	Maximum percentage of the Yarn disk quota that can be occupied by the container log directory on each disk. When the space occupied by the log directory exceeds the value of this parameter, the periodic log collection service is triggered to start a log collection activity beyond the period to release the local disk space. Maximum space for container logs that can be provided on each disk. If the disk space occupied by container logs exceeds this threshold, data aggregation in rolling mode is triggered. The valid value range of the maximum disk quota percentage is –1 to 100. If the value is less than –1, it is forcibly reset to **25**. If the value is greater than 100, the value is forcibly reset to **25**. If you set the value to **–1**, the disk capacity detection function for Container log directory is disabled. **NOTE** ● Percentage of the available disk space of the container log directory = Percentage of the available disk space of Yarn (**yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage**) x Percentage of the available disk space of the container log directory (**yarn.nodemanager.disk-health-checker.log-dirs.max-disk-utilization-per-disk-percentage**) ● Only applications with the periodic log collection function enabled can trigger log collection when the disk quota of the log directory exceeds the threshold.	25
yarn.nodemanager.remote-app-log-dir-suffix	Name of the HDFS folder in which container logs are to be stored. This parameter and **yarn.nodemanager.remote-app-log-dir** form the full path for storing container logs. That is, **{yarn.nodemanager.remote-app-log-dir}/${user}/{yarn.nodemanager.remote-app-log-dir-suffix}**. **NOTE** *{user}* indicates the username for running the task.	logs

Parameter	Description	Default Value
yarn.nodemanager.log-aggregator.on-fail.remain-log-in-sec	Duration for retaining container logs on the local host after the logs fail to be collected, in second • If this parameter is set to **0**, local logs are deleted immediately. • If this parameter is set to a positive number, local logs are retained for this period.	604800

Go to the **All Configurations** page of MapReduce and enter a parameter name in **Table 27-3** in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 27-3 Parameter description

Parameter	Description	Default Value
yarn.log-aggregation.retain-seconds	Duration for retaining aggregated logs, in second • If this parameter is set to **–1**, the container logs will be retained permanently in the HDFS. • If this parameter is set to **0** or a positive integer, container logs will be stored for such a period and deleted after the period expires. **NOTE** A short period may increase load of the NameNode. Therefore, you are advised to set this parameter to a proper value.	1296000

Parameter	Description	Default Value
yarn.log-aggregation.retain-check-interval-seconds	Interval for storing container logs in HDFS, in second • If this parameter is set to **-1** or **0**, the interval will be one tenth of the period specified by **yarn.log-aggregation.retain-seconds**. **NOTE** If this parameter is set to **-1** or **0**, **yarn.log-aggregation.retain-seconds** cannot be set to **0**. • If this parameter is set to a positive number, container logs in HDFS will be scanned at such an interval. **NOTE** A short interval may increase load of the NameNode. Therefore, you are advised to set this parameter to a proper value.	86400

Go to the **All Configurations** page of Yarn and enter a parameter name list in **Table 27-4** in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 27-4 Configuring rolling output of MapReduce application log files

Parameter	Description	Default Value
mapreduce.task.userlog.limit.kb	Maximum size of a single task log file of the MapReduce application. When the maximum size of the log file has been reached, a new log file is generated. The value **0** indicates that the size of the log file is not limited.	51200

Parameter	Description	Default Value
yarn.app.mapreduce.task.container.log.backups	Maximum number of task logs that can be retained for the MapReduce application. If this parameter is set to **0**, rolling output is disabled. Number of task log backup files when ContainerRollingLogAppender (CRLA) is used. By default, ContainerLogAppender (CLA) is used and container logs are not rolled back. When both **mapreduce.task.userlog.limit.kb** and **yarn.app.mapreduce.task.container.log.backups** are greater than 0, CRLA is enabled. The value ranges from 0 to 999.	10
yarn.app.mapreduce.am.container.log.limit.kb	Maximum size of a single ApplicationMaster log file of the MapReduce application, in KB. When the maximum size of the log file has been reached, a new log file is generated. The value **0** indicates that the size of a single ApplicationMaster log file is not limited.	51200
yarn.app.mapreduce.am.container.log.backups	Maximum number of ApplicationMaster logs that can be retained for the MapReduce application. If this parameter is set to **0**, rolling output is disabled. Number of ApplicationMaster log backup files when CRLA is used. By default, CLA is used and container logs are not rolled back. When both **yarn.app.mapreduce.am.container.log.limit.kb** and **yarn.app.mapreduce.am.container.log.backups** are greater than 0, CRLA is enabled for the ApplicationMaster. The value ranges from 0 to 999.	20
yarn.app.mapreduce.shuffle.log.backups	Maximum number of shuffle logs that can be retained for the MapReduce application. If this parameter is set to **0**, rolling output is disabled. When both **yarn.app.mapreduce.shuffle.log.limit.kb** and **yarn.app.mapreduce.shuffle.log.backups** are greater than 0, **syslog.shuffle** uses CRLA. The value ranges from 0 to 999.	10

Parameter	Description	Default Value
yarn.app.mapreduce.shuffle.log.limit.kb	Maximum size of a single shuffle log file of the MapReduce application, in KB. When the maximum size of the log file has been reached, a new log file is generated. If this parameter is set to **0**, the size of a single shuffle log file is not limited. The value must be greater than or equal to **0**.	51200

27.4 Enabling Yarn CGroups to Limit the Container CPU Usage

Scenario

CGroups is a Linux kernel feature. In YARN this feature allows containers to be limited in their resource usage (example, CPU usage). Without CGroups, it is hard to limit the container CPU usage. Without CGroups, it is hard to limit the container CPU usage.

NOTE

Currently, CGroups is only used for limiting the CPU usage.

Configuration Description

For details about how to configure CPU isolation and secure CGroups, visit the Hadoop official website:

Versions earlier than MRS 3.2.0: **http://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-site/NodeManagerCgroups.html**

MRS 3.2.0 or later: **https://hadoop.apache.org/docs/r3.3.1/hadoop-yarn/hadoop-yarn-site/NodeManagerCgroups.html**

CGroups is a Linux kernel feature and is enabled using LinuxContainerExecutor. For details about how to configure the LinuxContainerExecutor for security, see the official website. You can learn the file system permissions assigned to users and user groups from the official documentation at:

Versions earlier than MRS 3.2.0: **http://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-common/SecureMode.html#LinuxContainerExecutor**

MRS 3.2.0 or later: **https://hadoop.apache.org/docs/r3.3.1/hadoop-project-dist/hadoop-common/SecureMode.html#LinuxContainerExecutor**

NOTE

- Do not modify users, user groups, and related permissions of various paths in the corresponding file system. Otherwise, functions of CGroups may become abnormal.
- If the parameter value of **yarn.nodemanager.resource.percentage-physical-cpu-limit** is too small, the number of available cores may be less than one. For example, if the parameter of a four-core node is set to 20%, the number available core is less than one. As a result, all cores will be used. The Quota mode can be used in Linux versions, for example, Cent OS, that do not support Quota mode.

The table below describes the parameter for configuring cpuset mode, that is, only configured CPUs can be used by YARN.

Table 27-5 Parameter description

Parameter	Description	Default Value
yarn.nodemanager.linux-container-executor.cgroups.cpu-set-usage	Whether to enable the cpuset mode. If this parameter is set to **true**, the cpuset mode is enabled.	false

The table below describes the parameters for configuring the strictcpuset mode, that is, only configured CPUs can be used by containers.

Table 27-6 Parameter description

Parameter	Description	Default Value
yarn.nodemanager.linux-container-executor.cgroups.cpu-set-usage	Whether to enable the cpuset mode. If this parameter is set to **true**, the cpuset mode is enabled.	false
yarn.nodemanager.linux-container-executor.cgroups.cpuset.strict.enabled	Whether containers use allocated CPUs. If this parameter is set to **true**, the container can use the allocated CPUs.	false

To switch from cpuset mode to quota mode, the following conditions must be met:

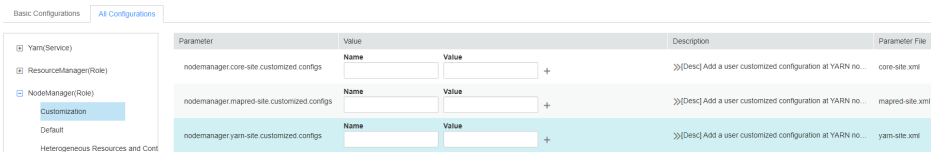
- Set the **yarn.nodemanager.linux-container-executor.cgroups.cpu-set-usage** parameter to **false**.
- Delete the **container** folder (if any) from the **/sys/fs/cgroup/cpuset/hadoop-yarn/** directory.
- Delete all CPUs configured in the **cpuset.cpus** file in **/sys/fs/cgroup/cpuset/hadoop-yarn/**.

Procedure

Step 1 Log in to Manager. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Yarn** > **Configurations** and select **All Configurations**.

Step 2 In the navigation pane on the left, choose **NodeManager** > **Customization** and find the **yarn-site.xml** file.

Step 3 Add the parameters in **Table 27-5** and **Table 27-6** as user-defined parameters.



Based on the configuration files and parameter functions, locate the row where parameter **yarn-site.xml** resides. Enter the parameter name in the **Name** column and enter the parameter value in the **Value** column.

Click **+** to add a customized parameter.

Step 4 Click **Save**. In the displayed **Save Configuration** dialog box, confirm the modification and click **OK**. Click **Finish** when the system displays "Operation succeeded". The configuration is successfully saved.

After the configuration is saved, restart the Yarn service whose configuration has expired for the configuration to take effect.

----End

27.5 Configuring HA for TimelineServer

Scenario

As a role of the Yarn service, TimelineServer supports the HA mode since the current version. To prevent a single point of failure of TimelineServer, you can enable TimelineServer HA to ensure high availability of the TimelineServer role.

NOTE

The TimelineServer saves data to the in-memory database LevelDB, which is memory-intensive. It is imperative to allocate a minimum of 30 GB of memory to the node hosting the TimelineServer.

This function applies to MRS 3.2.0-LTS.1 or later.

Impact on the System

- Before the conversion, change the value of **TLS_FLOAT_IP** of TimelineServer to an available floating IP address. (In the case of a single instance, the service IP address of the node is used by default.)
- During the conversion, the configuration of the TimelineServer role will expire. In this case, you need to restart the instance whose configuration expires.

Procedure

Step 1 Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Yarn**. Click **Configurations**.

Step 2 Change the value of **TLS_FLOAT_IP** to an available floating IP address (the floating IP address and the service IP addresses of the two TimelineServer instances must be in the same network segment) and click **Save** then **OK**.

Step 3 Click the **Instance** tab. Click **Add Instance**, select a node to add a TimelineServer instance, and choose **Next** > **Next** > **Submit**. The instance is added.

Step 4 On FusionInsight Manager, click ··· next to the cluster name, select **Restart Configuration-Expired Instances**, and wait until the instance is restarted.

Step 5 Check the status of each instance after the restart. For example, the active/standby status and running status of the TimelineServer instances are normal.

----End

27.6 Enterprise-Class Enhancements of Yarn

27.6.1 Configuring the Yarn Permission Control

Scenario

In the multi-tenant scenario in security mode, a cluster can be used by multiple users, and tasks of multiple users can be submitted and executed. Users are invisible to each other. A permission control mechanism is required to prevent task information of users from being obtained by other users.

For example, if user B logs in to the system and views the application list when the application submitted by user A is running, user B should not be able to view the application information of user A.

Configuration Description

- Viewing Yarn configuration parameters

  Go to the **All Configurations** page of Yarn and enter a parameter name list in **Table 27-7** in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

  Table 27-7 Parameter description

Parameter	Description	Default Value
yarn.acl.enable	Whether to enable Yarn permission control	true

Parameter	Description	Default Value
yarn.webapp.filter-entity-list-by-user	Whether to enable the strict view function. After this function is enabled, a login user can view only the content that the user has the permission to view. To enable this function, set **yarn.acl.enable** to **true**.	true

- Viewing MapReduce configuration parameters

  Go to the **All Configurations** page of MapReduce and enter a parameter name in **Table 27-8** in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

  Table 27-8 Parameter description

Parameter	Description	Default Value
mapreduce.cluster.acls.enabled	Whether to enable permission control of MapReduce JobHistoryServer This parameter is a client parameter and takes effect after permission control is enabled on the JobHistoryServer server.	true
yarn.webapp.filter-entity-list-by-user	Whether to enable the strict view of MapReduce JobHistoryServer. After the strict view is enabled, a login user can view only the content that the user has the permission to view. This parameter is a server parameter of JobHistoryServer. It indicates that permission control is enabled for JHS. However, whether to control a specific application is determined by the client parameter **mapreduce.cluster.acls.enabled**.	true

NOTICE

The preceding configurations affect the RESTful API and Shell command results. After the preceding configurations are enabled, the return results of RESTful API calls and shell commands contain only the information that the user has the permission to view.

If **yarn.acl.enable** or **mapreduce.cluster.acls.enabled** is set to **false**, the Yarn or MapReduce permission verification function is disabled. In this case, any user can submit tasks and view task information on Yarn or MapReduce, which poses security risks. Exercise caution when performing this operation.

27.6.2 Specifying the User Who Runs Yarn Tasks

Scenario

Currently, YARN allows the user that starts the NodeManager to run the task submitted by all other users, or the users to run the task submitted by themselves.

Configuration Description

On Manager, choose **Cluster** > *Name of the desired cluster* > **Services** > **Yarn** > **Configurations**. Click **All Configurations** Enter a parameter name in the search box.

Table 27-9 Parameter description

Parameter	Description	Default Value
yarn.nodemanager.linux-container-executor.user	Indicates the user who runs a task.	The value is left blank by default. **NOTE** The value is left blank by default. The user who submits a task is the actual person who runs the task.
yarn.nodemanager.container-executor.class	Indicates the executor who starts a task.	org.apache.hadoop.yarn.server.nodemanager.EnhancedLinuxContainerExecutor

NOTE

- Set **yarn.nodemanager.linux-container-executor.user** to configure the user who runs the container. This parameter is left blank by default. The user who submits the task is the person who runs the container. This parameter is valid only when **yarn.nodemanager.container-executor.class** is set to **org.apache.hadoop.yarn.server.nodemanager.EnhancedLinuxContainerExecutor**.
- In non-security mode, if **yarn.nodemanager.linux-container-executor.user** is set to **omm**, **yarn.nodemanager.linux-container-executor.nonsecure-mode.local-user** must also be set to **omm**.
- For security reasons, it is advised to retain the default values of **yarn.nodemanager.linux-container-executor.user** and **yarn.nodemanager.container-executor.class**.

27.6.3 Configuring the Number of ApplicationMaster Retries

Scenario

When resources are insufficient or ApplicationMaster fails to start, a client probably encounters running errors.

Configuration Description

Go to the **All Configurations** page of Yarn and enter a parameter name list in **Table 27-10** in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 27-10 Parameter description

Parameter	Description	Default Value
yarn.resourcemanager.am.max-attempts	Number of retries of the ApplicationMaster. Increasing the number of retries can prevent ApplicationMaster startup failures caused by insufficient resources. This applies to global settings of all ApplicationMasters. Each ApplicationMaster can use an API to set an independent maximum number of retries. However, the number of retries cannot be greater than the global maximum number of retries. If the value is greater than the global maximum number of retries, the ResourceManager overwrites the value to allow at least one retry. The value must be greater than or equal to **1**.	5

27.6.4 Configure the ApplicationMaster to Automatically Adjust the Allocated Memory

Scenario

During the process of starting the configuration, when the ApplicationMaster creates a container, the allocated memory is automatically adjusted according to the total number of tasks, which makes resource utilization more flexible and improves the fault tolerance of the client application.

Configuration Description

Navigation path for setting parameters:

On Manager, choose **Cluster** > *Name of the desired cluster* > **Service** > **Yarn** > **Configuration**. On the displayed page, select **All Configurations** and enter **mapreduce.job.am.memory.policy**.

Configuration description

If the default value of the parameter is left empty. In this case, the automatic adjustment policy is not enabled. The memory of ApplicationMaster is still affected by the value of **yarn.app.mapreduce.am.resource.mb**.

The value of **mapreduce.job.am.memory.policy** consists of five items, and they are separated by colons (:) and commas (,) in the following format: **baseTaskCount:taskStep:memoryStep,minMemory:maxMemory**. The format is strictly checked when the value is entered.

Table 27-11 Parameter description

Parameter	Description	Setting Requirement
baseTaskCount	Indicates the total number of tasks. The configuration of ApplicationMaster is valid only when the total number of tasks (on the sum of the Map and Reduce ends) is greater than or equal to the value of this parameter.	The value cannot be empty and must be greater than 0.
taskStep	Indicates the incremental step length of tasks. This parameter and **memoryStep** determine the memory adjustment amount.	The value cannot be empty and must be greater than 0.
memoryStep	Indicates the incremental memory step. The memory capacity is increased based on the value of **yarn.app.mapreduce.am.resource.mb**.	The value cannot be empty and must be greater than 0. The unit is MB.

Parameter	Description	Setting Requirement
minMemory	Indicates the lower limit of the memory that can be automatically adjusted. If the memory after the automatic adjustment is less than or equal to the value of this parameter, the value of **yarn.app.mapreduce.am.resource.mb** is used.	The value cannot be empty. It must be greater than 0 and cannot be greater than the value of **maxMemory**. Unit: MB
maxMemory	Indicates the upper limit of memory that can ve automatically adjusted. If the adjusted memory exceeds the upper limit, use this value as the final value.	The value cannot be empty. It must be greater than 0 and cannot be less than the value of **minMemory**. Unit: MB

Example Value

Configuration:

- yarn.app.mapreduce.am.resource.mb=1536
- mapreduce.job.am.memory.policy=100:10:50,1200:2000
- Total number of tasks of an application =120

The calculation process is as follows:

Memory after adjustment = 1536 + [(120 – 100)/10] x 50 = 1636. In this example, memory after adjustment 1636 is greater than the value of **minMemory 1200**, and less than the value of **maxMemory 2000**. Therefore, the ApplicationMaster memory is set to **1636 MB**.

If the value of **memStep** is changed to **250**, the calculation formula is as follows: Memory after adjustment = 1536 + [(120 – 100) / 10] x 250 = 2136. In this case, the memory after adjustment is greater than the value of **maxMemory 2000**. As a result, the value of **ApplicationMaster** is set to **2000 MB**.

NOTE

If the memory after adjustment is lower than the value of **minMemory**, the configuration does not take effect but the value is still printed on the backend server. This value is provided as the reference for adjusting the value of **minMemory**.

27.6.5 Configuring ApplicationMaster Work Preserving

Scenario

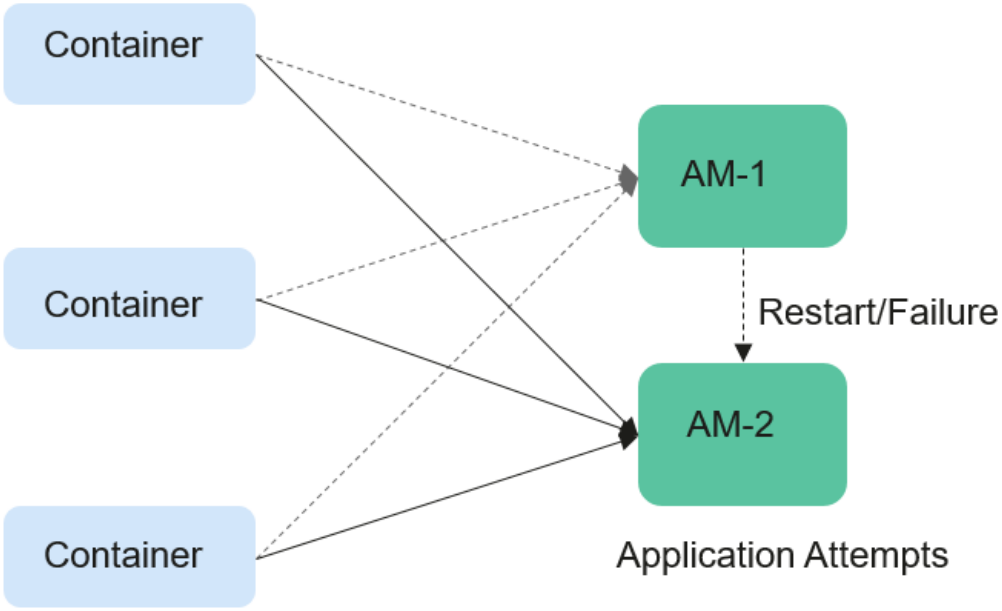
In YARN, ApplicationMasters run on NodeManagers just like every other container (ignoring unmanaged ApplicationMasters in this context). ApplicationMasters may break down, exit, or shut down. If an ApplicationMaster node goes down,

ResourceManager kills all the containers of ApplicationAttempt, including containers running on NodeManager. ResourceManager starts a new ApplicationAttempt node on another compute node.

For different types of applications, we want to handle ApplicationMaster restart events in different ways. MapReduce applications aim to prevent task loss but allow the loss of the currently running container. However, for the long-period YARN service, users may not want the service to stop due to the ApplicationMaster fault.

YARN can retain the status of the container when a new ApplicationAttempt is started. Therefore, running jobs can continue to operate without faults.

Figure 27-1 ApplicationMaster job preserving



Configuration Description

Go to the **All Configurations** page of Yarn and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Set the following parameters based on **Table 27-12**.

Table 27-12 Parameter description

Parameter	Description	Default Value
yarn.app.mapreduce.am.work-preserve	Whether to enable the ApplicationMaster job retention feature.	false
yarn.app.mapreduce.am.umbilical.max.retries	Maximum number of attempts to restore a running container in the ApplicationMaster job retention feature.	5

Parameter	Description	Default Value
yarn.app.mapreduce.am.umbilical.retry.interval	Specifies the interval at which a running container attempts to recover in the ApplicationMaster job retention feature. Unit: millisecond	10000
yarn.resourcemanager.am.max-attempts	The number of retries of ApplicationMaster. Increasing the number of retries prevents ApplicationMaster startup failures caused by insufficient resources. This applies to global settings of all ApplicationMasters. Each ApplicationMaster can use an API to set an independent maximum number of retries. However, the number of retries cannot be greater than the global maximum number of retries. If the value is greater than the global maximum number of retries, the ResourceManager overwrites the value The value must be greater than or equal to **1**.	2

27.6.6 Configuring the Access Channel Protocol

Scenario

The value of the **yarn.http.policy** parameter must be consistent on both the server and clients. Web UIs on clients will be garbled if an inconsistency exists, for example, the parameter value is **HTTPS_ONLY** on the server but it is left unspecified on a client (the parameter value **HTTP_ONLY** is applied to the client by default). Set the **yarn.http.policy** parameters on the clients and server to prevent garbled characters from being displayed on the clients.

Procedure

Step 1 On Manager, choose **Cluster** > *Name of the desired cluster* > **Services** > **Yarn** > **Configurations**. On the displayed page, select **All Configurations** and enter **yarn.http.policy**.

- In security mode, set this parameter to **HTTPS_ONLY**.
- In normal mode, set this parameter to **HTTP_ONLY**.

Step 2 Log in to the node where the client is installed as the client installation user.

Step 3 Run the following command to switch to the client installation directory:

cd /opt/client

Step 4 Run the following command to edit the **yarn-site.xml** file:

vi Yarn/config/yarn-site.xml

Change the value of **yarn.http.policy**.

In security mode, set this parameter to **HTTPS_ONLY**.

In normal mode, set this parameter to **HTTP_ONLY**.

Step 5 Run the **:wq** command to save execution.

Step 6 Restart the client for the settings to take effect.

----End

27.6.7 Configuring the Additional Scheduler WebUI

Scenario

If the custom scheduler is set in ResourceManager, you can set the corresponding web page and other Web applications for the custom scheduler.

Configuration Description

Go to the **All Configurations** page of Yarn and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 27-13 Configuring the Additional Scheduler WebUI

Parameter	Description	Default Value
hadoop.http.rmwebapp.scheduler.page.classes	Load the corresponding web page for the custom scheduler on the RM WebUI. This parameter is valid only when **yarn.resourcemanager.scheduler.class** is set to a custom scheduler.	-
yarn.http.rmwebapp.external.classes	Load the custom web application in the RM Web service.	-

27.6.8 Configuring Resources for a NodeManager Role Instance

Scenario

If the hardware resources (such as the number of CPU cores and memory size) of the nodes for deploying NodeManagers are different but the NodeManager available hardware resources are set to the same value, the resources may be wasted or the status may be abnormal. You need to change the hardware resource configuration for each NodeManager to ensure that the hardware resources can be fully utilized.

Impact on the System

NodeManager role instances must be restarted for the new configuration to take effect, and the role instances are unavailable during restart.

Prerequisites

You have logged in to Manager.

Procedure

Step 1 Choose **Cluster** > *Name of the desired cluster* > **Services** > **Yarn** > **Instance**.

Step 2 Click the role instance name corresponding to the node where NodeManager is deployed, switch to **Instance Configuration**, and select **All Configurations**.

Step 3 Enter **yarn.nodemanager.resource.cpu-vcores** in the search box, and set the number of vCPUs that can be used by NodeManager on the current node. You are advised to set this parameter to 1.5 to 2 times the number of actual logical CPUs on the node. Enter **yarn.nodemanager.resource.memory-mb** in the search box, and set the physical memory size that can be used by NodeManager on the current node. You are advised to set this parameter to 75% of the actual physical memory size of the node.

NOTE

Enter **yarn.scheduler.maximum-allocation-vcores** in the search box, and set the maximum number of available CPUs in a container. Enter **yarn.scheduler.maximum-allocation-mb** in the search box, and set the maximum available memory of a container. The instance level cannot be changed. The parameter values need to be changed in the configuration of the Yarn service, and the Yarn service needs to be restarted for the changes to take effect.

Step 4 Click **Save**, and then click **OK**. to restart the NodeManager role instance.

A message is displayed, indicating that the operation is successful. Click **Finish**. The NodeManager role instance is started successfully.

----End

27.6.9 Configuring Yarn Restart

Scenario

The Yarn Restart feature includes ResourceManager Restart and NodeManager Restart.

- When ResourceManager Restart is enabled, the new active ResourceManager node loads the information of the previous active ResourceManager node, and takes over container status information on all NodeManager nodes to continue service running. In this way, status information can be saved by periodically executing checkpoint operations, avoiding data loss.
- When NodeManager Restart is enabled, NodeManager locally saves information about containers running on the node. After NodeManager is restarted, the container running progress on the node will not be lost by restoring the saved status information.

Configuration Description

Go to the **All Configurations** page of Yarn and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Configure ResourceManager Restart as follows:

Table 27-14 Parameter description of ResourceManager Restart

Parameter	Description	Default Value
yarn.resourcemanager.recovery.enabled	Whether to enable ResourceManager to restore the status after startup. If this parameter is set to **true**, **yarn.resourcemanager.store.class** must also be set.	true
yarn.resourcemanager.store.class	State-store class used to store the application and task statuses and certificate content.	org.apache.hadoop.yarn.server.resourcemanager.recovery.AsyncZKRMStateStore
yarn.resourcemanager.zk-state-store.parent-path	Directory for storing ZKRMStateStore in ZooKeeper	/rmstore
yarn.resourcemanager.work-preserving-recovery.enabled	Whether to enable ResourceManager work serving. This configuration is used only for Yarn feature verification.	true
yarn.resourcemanager.state-store.async.load	Whether to apply asynchronous restoration to completed applications.	**true**
yarn.resourcemanager.zk-state-store.num-fetch-threads	If asynchronous restoration is enabled, increasing the number of working threads can speed up the restoration of task information stored in ZooKeeper. The value must be greater than 0.	**20**

Configure NodeManager Restart as follows:

Table 27-15 Parameter description of NodeManager Restart

Parameter	Description	Default Value
yarn.nodemanager.recovery.enabled	Whether to enable the function of collecting logs upon a log collection failure when NodeManager is restarted and whether to restore the unfinished application	true
yarn.nodemanager.recovery.dir	Local directory used by NodeManager to store container status	${SRV_HOME}/tmp/yarn-nm-recovery
yarn.nodemanager.recovery.supervised	Whether NodeManager is monitored. After this parameter is enabled, NodeManager does not clear containers after exit. NodeManager assumes that it will restart and restore containers immediately.	true

27.7 Yarn Performance Tuning

27.7.1 Preempting a Task

Scenario

- **Preemption principle of the Capacity scheduler:**

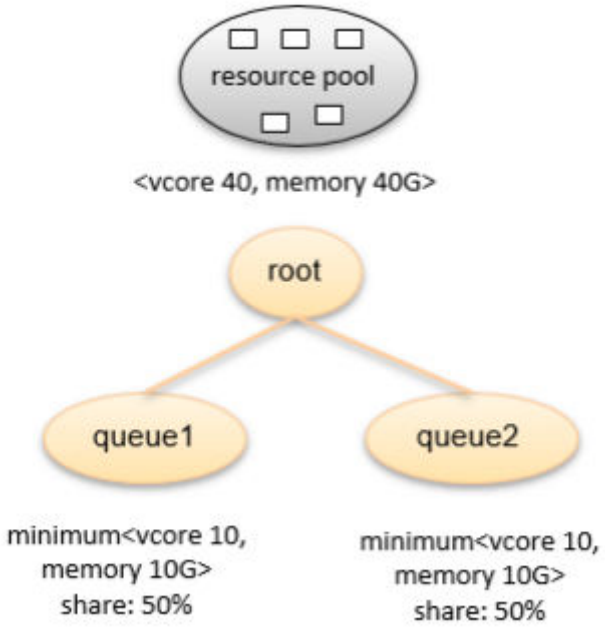
  The capacity scheduler of ResourceManager implements job preemption to simplify job running in queues and improve resource utilization. The process is as follows:

  a. Assume that there are two queues (Queue A and Queue B). The capacity of Queue A is 25%, and the capacity of Queue B is 75%.
  b. In the initial state, Task 1 is distributed to Queue A for processing, requiring 75% cluster resources. Task 2 is distributed to Queue B for processing, requiring 50% cluster resources.
  c. Task 1 uses 25% cluster resources provided by Queue A and 50% resources from Queue B. Queue B reserves 25% cluster resources.
  d. If task preemption is enabled, the resources of Task 1 will be preempted. Queue B preempts 25% cluster resources from Queue A for Task 2.
  e. Task 1 will be executed when Task 2 is complete and the cluster has sufficient resources.

- **Preemption Principle of the Superior Scheduler**

  When cluster resources are abundant, the Superior Scheduler permits queues to utilize resources beyond their minimum and shared allocations, improving

overall resource utilization. Conversely, when resources are scarce, preemption ensures that each queue receives its minimum guaranteed resources as per the queue policy, maintaining fairness. The following are examples: The queue resource information is as follows (**minimum** indicates the minimum resource, and **share** indicates the weight):



a. User A submits Job 1 to Queue 1. The Application Master (AM) allocates 1 vCore and 2 GB memory for itself and initiates 8 tasks, each consuming 2 vCores and 4 GB memory. Given that resources are sufficient, Job 1 executes smoothly. Queue1, in total, utilizes 17 vCores and 34 GB memory. Subsequently, the cluster's remaining resources tally at 23 vCores and 6 GB memory.
b. User B submits Job 2 to Queue 2. The AM allocates 1 vCore and 2 GB memory to itself and initiates 5 tasks, each task consuming 2 vCores and 4 GB memory. Prior to any preemption, both the AM of Job 2 and one of its tasks operate seamlessly. Queue 2 consumes 3 vCores and 6 GB memory, leaving the cluster with no available memory and merely 20 vCores.
c. Since Queue2 has yet to utilize its minimum guaranteed resources and has pending tasks, the Scheduler initiates preemption to reclaim resources from a task of Job 1. This action allows Job 2 to deploy an additional task using the recaptured resources. After preemption, Queue 2's resource usage increases to 5 vCores and 10 GB memory.

NOTE

If either CPU resources or memory resources of a queue are its minimum guaranteed resources, the queue resources will not be preempted by other queues.

Procedure

Navigation path for setting parameters:

Go to the **All Configurations** page of Yarn and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 27-16 Parameter description

Parameter	Description	Default Value
yarn.resourcemanager.scheduler.monitor.enable	Whether to start scheduler monitoring according to **yarn.resourcemanager.scheduler.monitor.policies**. If this parameter is set to **true**, scheduler monitoring is enabled based on policies specified by **yarn.resourcemanager.scheduler.monitor.policies** and task resource preemption is enabled based on the scheduler information. If this parameter is set to **false**, scheduler monitoring is disabled.	false
yarn.resourcemanager.scheduler.monitor.policies	List of the SchedulingEditPolicy class to be used with the scheduler	org.apache.hadoop.yarn.server.resourcemanager.monitor.capacity.ProportionalCapacityPreemptionPolicy
yarn.resourcemanager.monitor.capacity.preemption.observe_only	• If this parameter is set to **true**, policies will be applied but task resource preemption will not be performed. • If this parameter is set to **false**, policies will be applied and task resource preemption will be performed based on the policies.	false
yarn.resourcemanager.monitor.capacity.preemption.monitoring_interval	Monitoring interval, in millisecond. If this parameter is set to a larger value, capacity detection will not be performed frequently.	3000
yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill	Interval between the time when a resource preemption request is sent and the time when the container is stopped (resources are released), in millisecond. The value must be greater than or equal to **0**. By default, if ApplicationMaster does not stop the container within 15 seconds, ResourceManager will forcibly stop the container after 15 seconds.	15000

Parameter	Description	Default Value
yarn.resourcemanager.monitor.capacity.preemption.total_preemption_per_round	Maximum resource preemption ratio in a period. This value can be used to limit the speed at which containers are reclaimed from the cluster. After the expected total preemption value is calculated, the policy scales the preemption ratio back to this limit.	0.1
yarn.resourcemanager.monitor.capacity.preemption.max_ignored_over_capacity	Resource preemption dead zone = Total number of resources in the cluster x Value of this configuration item + Original resources of a queue (for example, Queue A). When resources actually used by a task in Queue A exceeds the preemption dead zone, the resource beyond the preemption dead zone is preempted. The value range is 0 to 1. **NOTE** A smaller value is recommended for effective preemption.	0
yarn.resourcemanager.monitor.capacity.preemption.natural_termination_factor	Preemption percentage. Containers preempt only this percentage of the resources. For example, a termination factor of 0.5 will reclaim almost 95% of resources within 5 times of **yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill**, even in the absence of natural termination. That is, 5 consecutive preemptions will be performed and each time half of the target resources will be preempted. The trend is geometric convergence. The interval of each preemption is **yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill**. The value range is 0 to 1.	1
ss.engine.scheduler.preemption-enable	Whether to enable Superior scheduler preemption.	false
ss.engine.scheduler.preemption-interval-ms	Minimum interval for triggering preemption (also called preemption period). The unit is millisecond.	3000

Parameter	Description	Default Value
ss.engine.scheduler.preemption-max-per-interval	Maximum number of YARN containers that can be preempted within a preemption period. The default value is **-1**, indicating that there is no restriction on the maximum number of YARN containers.	-1
ss.engine.scheduler.preemption-warn-period-ms	Preemption duration to be notified to AM. Should the AM fail to relinquish the borrowed resources within the stipulated timeframe, the YARN container employing those resources will be forcibly terminated by the Resource Manager. The unit is millisecond.	10000

27.7.2 Setting the Task Priority

Scenario

The resource contention scenarios of a cluster are as follows:

1. Submit two jobs (Job 1 and Job 2) with lower priorities.
2. Some tasks of running Job 1 and Job 2 are in the running state. However, some tasks are pending due to resource deficiency because the capacity of cluster or queue resources is limited.
3. Submit a job (Job 3) with a higher priority. In this case, after the running tasks of Job 1 and Job 2 are complete, their resources will be released and then allocated to the pending tasks of Job 3.
4. After Job 3 is complete, its resources will be released and then allocated to Job 1 and Job 2.

Users can use capacity scheduler of ResourceManager to set the task priority in Yarn because the task priority is implemented by the scheduler of ResourceManager.

Procedure

Set the **mapreduce.job.priority** parameter and use CLI or API to set the task priority.

- Through the CLI

  When submitting tasks, add the **-Dmapreduce.job.priority=***<priority>* parameter.

  <priority> can be set to any of the following values:

  – VERY_HIGH

  - HIGH
  - NORMAL
  - LOW
  - VERY_LOW
- Through the API

  You can also set the task priority through the API.

  Set **Configuration.set("mapreduce.job.priority", <priority>)** or **Job.setPriority(JobPriority priority)**.

27.7.3 Optimizing Node Configuration

Scenario

After the scheduler of a big data cluster is properly configured, you can adjust the available memory, CPU resources, and local disk of each node to optimize the performance.

The configuration items are as follows:

- Available memory
- Number of vCPUs
- Physical CPU usage
- Coordination of memory and CPU resources
- Local disk

Procedure

For details about how to adjust parameter settings, see **Modifying Cluster Service Configuration Parameters**.

- **Available memory**

  Except the memory allocated to the OS and other services, allocate as much as possible memory to Yarn. You can adjust the following parameters to improve resource utilization.

  Assume that a container uses 512 MB memory by default, then the memory usage formula is: 512 MB x Number of containers.

  By default, the Map or Reduce container uses one vCPU and 1,024 MB memory, and ApplicationMaster uses 1,536 MB memory.

Parameter	Description	Default Value
yarn.nodemanager.resource.memory-mb	Physical memory that can be allocated to containers, in MB. The value must be greater than 0. You are advised to set the parameter value to 75% to 90% of the total physical memory of nodes. If the node has permanent processes of other services, reduce this parameter value to reserve sufficient resources for the processes.	**16384**

- **Number of vCPUs**

  You are advised to set this parameter to 1.5 to 2 times the number of logical CPUs. If the upper layer computing applications have low computing capability requirements, you can set the parameter to two times the number of logical CPUs.

Parameter	Description	Default Value
yarn.nodemanager.resource.cpu-vcores	Number of vCPUs that can be used by Yarn on the node. The default value is **8**. You are advised to set the value to 1.5 to 2 times the number of logical CPUs.	8

- **Physical CPU usage**

  You are advised to reserve appropriate CPUs for the OS and the processes, such as database and HBase, and allocate the remaining CPUs to Yarn. You can set the following parameters to adjust the physical CPU usage.

Parameter	Description	Default Value
yarn.nodemanager.resource.percentage-physical-cpu-limit	Physical CPU percentage that can be used by Yarn on a node. The default value is **90**, indicating that no CPU control is implemented and Yarn can use all CPU resources. You can only view the parameter. To change the value of this parameter, set the value of RES_CPUSET_PERCENTAGE of YARN. You are advised to set this parameter to the percentage of CPU resources that can be used by the YARN cluster. For example, If 20% of CPU resources are used by other services (such as HBase, HDFS, and Hive) and system processes on the node, the CPU resources can be scheduled for Yarn is 1 - 20% = 80%. Therefore, you can set this parameter to **80**.	90

- **Local disk**

  MapReduce writes the intermediate job execution results in local disks. Therefore, configure disks as much as possible and disk space as large as possible. A simple way is to configure the same number of disks as DataNode except for the last directory.

  NOTE

  Use commas (,) to separate multiple disks.

Parameter	Description	Default Value
yarn.nodemanager.log-dirs	Directories in which logs are stored. Multiple directories can be specified. Storage location of container logs. The default value is **%{@auto.detect.datapart.nm.logs}**. If there is a data partition, a path list similar to **/srv/BigData/hadoop/data1/nm/containerlogs,/srv/BigData/hadoop/data2/nm/containerlogs** is generated based on the data partition. If there is no data partition, the default path **/srv/BigData/yarn/data1/nm/containerlogs** is generated. In addition to using expressions, you can enter a complete list of paths, such as **/srv/BigData/yarn/data1/nm/containerlogs** or **/srv/BigData/yarn/data1/nm/containerlogs,/srv/BigData/yarn/data2/nm/containerlogs**. In this way, data is stored in all the configured directories, which are usually on different devices. To ensure disk I/O load balancing, you are advised to provide several paths and each path corresponds to an independent disk. The localized log directory of the application exists in the relative path **/application_%{appid}**. The log directory of an independent container, that is, **container_{$contid}**, is the subdirectory of this directory. Each container directory contains the **stderr**, **stdin**, and **syslog** files generated by the container. To add a directory, for example, **/srv/BigData/yarn/data2/nm/containerlogs**, you need to delete the files in **/srv/BigData/yarn/data2/nm/containerlogs** first. Then, assign the same read and write permissions to **/srv/BigData/yarn/data2/nm/containerlogs** as those of **/srv/**	%{@auto.detect.datapart.nm.logs}

Parameter	Description	Default Value
	BigData/yarn/data1/nm/containerlogs, and change **/srv/BigData/yarn/data1/nm/containerlogs** to **/srv/BigData/yarn/data1/nm/containerlogs,/srv/BigData/yarn/data2/nm/containerlogs**. You can add directories, but do not modify or delete existing directories. Otherwise, NodeManager data will be lost and services will be unavailable. Default value: **%{@auto.detect.datapart.nm.logs}** Exercise caution when modifying this parameter. If the configuration is incorrect, the services are unavailable. If the value of this configuration item at the role level is changed, the value of this configuration item at all instance levels will be changed. If the value of this configuration item at the instance level is changed, the value of this configuration item of other instances remains unchanged.	

Parameter	Description	Default Value
yarn.nodemanager.local-dirs	Storage location of files after localization. The default value is **%{@auto.detect.datapart.nm.localdir}**. If there is a data partition, a path list similar to **/srv/BigData/hadoop/data1/nm/localdir,/srv/BigData/hadoop/data2/nm/localdir** is generated based on the data partition. If there is no data partition, the default path **/srv/BigData/yarn/data1/nm/localdir** is generated. In addition to using expressions, you can enter a complete list of paths, such as **/srv/BigData/yarn/data1/nm/localdir** or **/srv/BigData/yarn/data1/nm/localdir,/srv/BigData/yarn/data2/nm/localdir**. In this way, data is stored in all the configured directories, which are usually on different devices. To ensure disk I/O load balancing, you are advised to provide several paths and each path corresponds to an independent disk. The localized file directory of the application is stored in the relative path **/usercache/%{user}/appcache/application_%{appid}**. The working directory of an independent container, that is, **container_%{contid}**, is the subdirectory of the directory. To add a directory, for example, **/srv/BigData/yarn/data2/nm/localdir**, you need to delete the files in **/srv/BigData/yarn/data2/nm/localdir** first. Then, assign the same read and write permissions to **/srv/BigData/hadoop/data2/nm/localdir** as those of **/srv/BigData/hadoop/data1/nm/localdir**, and change **/srv/BigData/yarn/data1/nm/localdir** to **/srv/BigData/yarn/data1/nm/localdir,/srv/BigData/yarn/data2/nm/localdir**. You can add	%{@auto.detect.datapart.nm.localdir}

Parameter	Description	Default Value
	directories, but do not modify or delete existing directories. Otherwise, NodeManager data will be lost and services will be unavailable. Default value: **%{@auto.detect.datapart.nm.localdir}** Exercise caution when modifying this parameter. If the configuration is incorrect, the services are unavailable. If the value of this configuration item at the role level is changed, the value of this configuration item at all instance levels will be changed. If the value of this configuration item at the instance level is changed, the value of this configuration item of other instances remains unchanged.	

27.8 Yarn O&M Management

27.8.1 YARN Common Configuration Parameters

Allocating Queue Resources

The Yarn service provides queues for users. Users allocate system resources to each queue. After the configuration is complete, you can click **Refresh Queue** or restart the Yarn service for the configuration to take effect.

Navigation path for setting parameters:

On Manager, choose **Tenant Resources** > **Dynamic Resource Plan** > **Queue Configuration**.

The following uses the **default** tenant who modifies the Superior scheduler as an example. The configurations of other queues are similar. Click **Modify** to edit the parameters.

Table 27-17 Queue configuration parameters

Parameter	Description
Max Master Shares(%)	Indicates the maximum percentage of resources occupied by all ApplicationMasters in the current queue.
Max Allocated vCores	Indicates the maximum number of cores that can be allocated to a single YARN container in the current queue. The default value is **-1**, indicating that the number of cores is not limited within the value range.
Max Allocated Memory(MB)	Indicates the maximum memory that can be allocated to a single YARN container in the current queue. The default value is **-1**, indicating that the memory is not limited within the value range.
Max Running Apps	Maximum number of tasks that can be executed at the same time in the current queue. The default value is **-1**, indicating that the number is not limited within the value range (the meaning is the same if the value is empty). The value 0 indicates that the task cannot be executed. The value ranges from -1 to 2147483647.
Max Running Apps per User	Maximum number of tasks that can be executed by each user in the current queue at the same time. The default value is **-1**, indicating that the number is not limited within the value range. If the value is **0**, the task cannot be executed. The value ranges from -1 to 2147483647.
Max Pending Apps	Maximum number of tasks that can be suspended at the same time in the current queue. The default value is **-1**, indicating that the number is not limited within the value range (the meaning is the same if the value is empty). The value **0** indicates that tasks cannot be suspended. The value ranges from -1 to 2147483647.
Resource Allocation Rule	Indicates the rule for allocating resources to different tasks of a user. The rule can be FIFO or FAIR. If a user submits multiple tasks in the current queue and the rule is FIFO, the tasks are executed one by one in sequential order; if the rule is FAIR, resources are evenly allocated to all tasks.
Default Resource Label	Indicates that tasks are executed on a node with a specified resource label.
Active	• **ACTIVE**: indicates that the current queue can receive and execute tasks. • **INACTIVE**: indicates that the current queue can receive but cannot execute tasks. Tasks submitted to the queue are suspended.

Parameter	Description
Open	• **OPEN**: indicates that the current queue is opened. • **CLOSED**: indicates that the current queue is closed. Tasks submitted to the queue are rejected.

Displaying Container Logs on the Web UI

By default, the system collects container logs to HDFS. If you do not need to collect container logs to HDFS, configure the parameters in **Table 27-18**. For details, see **Modifying Cluster Service Configuration Parameters**.

Table 27-18 Parameter description

Parameter	Description	Default Value
yarn.log-aggregation-enable	Select whether to collect container logs to HDFS. • If the parameter is set to **true**, container logs are collected to an HDFS directory. The default directory is **{yarn.nodemanager.remote-app-log-dir}/${user}/{thisParam}**. You can set the directory by setting the **yarn.nodemanager.remote-app-log-dir-suffix** parameter on the web UI. • If this parameter is set to **false**, container logs will not be collected to HDFS. After changing the parameter value, restart the Yarn service for the setting to take effect. **NOTE** The container logs that are generated before the parameter is set to **false** and the setting takes effect cannot be obtained from the web UI. You can obtain container logs from the directory specified by the **yarn.nodemanager.remote-app-log-dir-suffix** parameter before the setting takes effect. If you want to view the logs generated before on the web UI, you are advised to set this parameter to **true**.	true

Increasing the Number of Historical Jobs to Be Displayed on the web UI

By default, the Yarn web UI supports task list pagination. A maximum of 5,000 historical jobs can be displayed on each page, and a maximum of 10,000 historical jobs can be retained. If you need to view more jobs on the WebUI, configure

parameters by referring to **Table 27-19**. For details, see **Modifying Cluster Service Configuration Parameters**.

Table 27-19 Parameter description

Parameter	Description	Default Value
yarn.resourcemanager.max-completed-applications	Set the total number of historical jobs to be displayed on the web UI.	10000
yarn.resourcemanager.webapp.pagination.enable	Select whether to enable the job list background pagination function for the Yarn web UI.	true
yarn.resourcemanager.webapp.pagination.threshold	Set the maximum number of jobs displayed on each page after the job list background pagination function of the Yarn web UI is enabled.	5000

NOTE

- If a large number of historical jobs are displayed, the performance will be affected and the time for opening the Yarn web UI will be increased. Therefore, you are advised to enable the background pagination function and modify the **yarn.resourcemanager.max-completed-applications** parameter according to the actual hardware performance.
- After changing the parameter value, restart the Yarn service for the setting to take effect.

27.8.2 Yarn Log Overview

Log Description

The default paths for saving Yarn logs are as follows:

- ResourceManager: **/var/log/Bigdata/yarn/rm** (run logs) and **/var/log/Bigdata/audit/yarn/rm** (audit logs)
- NodeManager: **/var/log/Bigdata/yarn/nm** (run logs) and **/var/log/Bigdata/audit/yarn/nm** (audit logs)

Log archive rule: The automatic compression and archive function is enabled for Yarn logs. By default, when the size of a log file exceeds 50 MB, the log file is automatically compressed. The naming rule of the compressed log file is as follows: *<Original log file name>-<yyyy-mm-dd_hh-mm-ss>.[ID]***.log.zip**. A maximum of 100 latest compressed files are retained. The number of compressed files can be configured on Manager.

Log archive rule:

Table 27-20 Yarn log list

Log Type	Log File Name	Description
Run log	hadoop-<SSH_USER>-<process_name>-<hostname>.log	Yarn component log file that records most of the logs generated when the Yarn component is running
	hadoop-<SSH_USER>-<process_name>-<hostname>.out	Log file that records Yarn running environment information
	<process_name>-<SSH_USER>-<DATE>-<PID>-gc.log	Garbage collection log file
	yarn-haCheck.log	ResourceManager active/standby status detection log file
	yarn-service-check.log	Log file that records the health check details of the Yarn service
	yarn-start-stop.log	Log file that records the startup and stop of the Yarn service
	yarn-prestart.log	Log file that records cluster operations before the Yarn service startup
	yarn-postinstall.log	Work log file after installation and before startup of the Yarn service
	hadoop-commission.log	Yarn service entry log file
	yarn-cleanup.log	Log file that records the cleanup operation during uninstallation of the Yarn service
	yarn-refreshqueue.log	Yarn queue refresh log file
	upgradeDetail.log	Upgrade log file
	stderr/stdin/syslog	Container log file of the applications running on the Yarn service
	yarn-application-check.log	Check log file of applications running on the Yarn service

Log Type	Log File Name	Description
	yarn-appsummary.log	Running result log file of applications running on the Yarn service
	yarn-switch-resourcemanager.log	Run log file that records the Yarn active/standby switchover
	ranger-yarn-plugin-enable.log	Log file that records the enabling of Ranger authentication for Yarn
	yarn-nodemanager-period-check.log	Periodic check log of Yarn NodeManager
	yarn-resourcemanager-period-check.log	Periodic check log of Yarn ResourceManager
	hadoop.log	Hadoop client logs
	env.log	Environment information log file before the instance is started or stopped.
Audit logs	yarn-audit-<process_name>.log ranger-plugin-audit.log	Yarn operation audit log file
	SecurityAuth.audit	Yarn security audit log file

Log Level

Table 27-21 describes the log levels supported by Yarn, including OFF, FATAL, ERROR, WARN, INFO, and DEBUG, from high priority to low. Logs whose levels are higher than or equal to the specified level are printed. The number of printed logs decreases as the specified log level increases.

Table 27-21 Log levels

Level	Description
FATAL	Logs of this level record critical error information about the current event processing.
ERROR	Logs of this level record error information about the current event processing.
WARN	Logs of this level record exception information about the current event processing.

Level	Description
INFO	Logs of this level record normal running status information about the system and events.
DEBUG	Logs of this level record the system as well as system debugging information.

To modify log levels, perform the following operations:

Step 1 Go to the **All Configurations** page of the Yarn service by referring to **Modifying Cluster Service Configuration Parameters**.

Step 2 On the menu bar on the left, select the log menu of the target role.

Step 3 Select a desired log level.

Step 4 Click **Save Configuration**. In the dialog box that is displayed, click **OK** to make the setting take effect.

NOTE

The configurations take effect immediately without the need to restart the service.

----End

Log Format

The following table lists the Yarn log formats.

Table 27-22 Log formats

Log Type	Format	Example
Run log	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<*Thread that generates the log*>|<*Message in the log*>|<*Location of the log event*>	2021-09-26 14:18:59,109 | INFO | main | Client environment:java.compiler=<NA> | org.apache.zookeeper.Environment.logEnv(Environment.java:100)
Audit log	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<*Thread that generates the log*>|<*Message in the log*>|<*Location of the log event*>	2021-09-26 14:24:43,605 | INFO | main-EventThread | USER=omm OPERATION=refreshAdminAcls TARGET=AdminService RESULT=SUCCESS | org.apache.hadoop.yarn.server.resourcemanager.RMAuditLogger$LogLevel$6.printLog(RMAuditLogger.java:91)

27.8.3 Configuring the Localized Log Levels

Scenarios

The default log level of localized container is **INFO**. You can change the log level by configuring **yarn.nodemanager.container-localizer.java.opts**.

Configuration Description

On Manager, choose **Cluster** > *Name of the desired cluster* > **Service** > **Yarn** > **Configuration**. Select **All Configurations** and set the following parameters in the configuration file **yarn-site.xml** of NodeManager to change the log level.

Table 27-23 Parameter description

Parameter	Description	Default Value
yarn.nodemanager.container-localizer.java.opts	The additional **jvm** parameters are provided for the localized container process.	-Xmx256m -Djava.security.krb5.conf=${KRB5_CONFIG}

The default value is **-Xmx256m -Djava.security.krb5.conf=${KRB5_CONFIG}** and the default log level is info. To change the localized log level of the container, add the following content:

```
-Dhadoop.root.logger=<LOG_LEVEL>,localizationCLA
```

Example:

To change the local log level to **DEBUG**, set the parameter as follows:

```
-Xmx256m -Dhadoop.root.logger=DEBUG,localizationCLA
```

NOTE

Allowed log levels are as follows: FATAL, ERROR, WARN, INFO, DEBUG, TRACE, and ALL.

27.8.4 Configuring Memory Usage Detection

Scenario

If memory usage of the submitted application cannot be estimated, you can modify the configuration on the server to determine whether to check the memory usage.

If the memory usage is not checked, the container occupies the memory until the memory overflows. If the memory usage exceeds the configured memory size, the corresponding container is killed.

Configuration Description

Go to the **All Configurations** page of Yarn and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 27-24 Parameter description

Parameter	Description	Default Value
yarn.nodemanager.vmem-check-enabled	Whether to enable virtual memory usage detection. If the memory used by a task exceeds the allocated memory size, the task is forcibly stopped. ● If the value is **true**, the virtual memory will be checked. ● If the value is **false**, the virtual memory will not be checked.	true
yarn.nodemanager.pmem-check-enabled	Whether to enable physical memory usage detection. If the memory used by a task exceeds the allocated memory size, the task is forcibly stopped. ● If the value is **true**, the physical memory will be checked. ● If the value is **false**, the physical memory will not be checked.	true

27.8.5 Changing NodeManager Storage Directories

Scenario

If the storage directories defined by YARN NodeManager are incorrect or the YARN storage plan changes, the MRS cluster administrator needs to modify the NodeManager storage directories on FusionInsight Manager to ensure smooth YARN running. The storage directories of NodeManager include the local storage directory **yarn.nodemanager.local-dirs** and log directory **yarn.nodemanager.log-dirs**. Changing the ZooKeeper storage directory includes the following scenarios:

- Change the storage directory of the NodeManager role. In this way, the storage directories of all NodeManager instances are changed.
- Change the storage directory of a single NodeManager instance. In this way, only the storage directory of this instance is changed, and the storage directories of other instances remain the same.

Impact on the System

- The cluster needs to stopped and restarted during the process of changing the storage directory of the NodeManager role, and the cluster cannot provide services before started.
- The NodeManager instance needs to stopped and restarted during the process of changing the storage directory of the instance, and the instance at this node cannot provide services before it is started.
- The directory for storing service parameter configurations must also be updated.

- After the storage directories of NodeManager are changed, you need to download and install the client again.

Prerequisites

- New disks have been prepared and installed on each data node, and the disks are formatted.
- New directories have been planned for storing data in the original directories.
- The MRS cluster administrator user **admin** has been prepared.

Procedure

Step 1 Check the environment.

1. Log in to Manager, choose **Cluster** > *Name of the desired cluster* > **Service** to check whether **Running Status** of Yarn is **Normal**.
   - If yes, go to **1.c**.
   - If no, the Yarn status is unhealthy. In this case, go to **1.b**.
2. Rectify faults of Yarn. No further action is required.
3. Determine whether to change the storage directory of the NodeManager role or that of a single NodeManager instance:
   - To change the storage directory of the NodeManager role, go to **2**.
   - To change the storage directory of a single NodeManager instance, go to **3**.

Step 2 Change the storage directory of the NodeManager role.

1. Choose **Cluster** > *Name of the desired cluster* > **Service** > **Yarn** > **Stop** to stop the Yarn service.
2. Log in to each data node where the Yarn service is installed as user **root** and perform the following operations:
   a. Create a target directory.

      For example, to create the target directory **${BIGDATA_DATA_HOME}/data2**, run the following command:

      mkdir ${BIGDATA_DATA_HOME}/data2

   b. Mount the target directory to the new disk.

      For example, mount **${BIGDATA_DATA_HOME}/data2** to the new disk.

   c. Modify permissions on the new directory.

      For example, to modify permissions on the **${BIGDATA_DATA_HOME}/data2** directory, run the following commands:

      chmod 750 ${BIGDATA_DATA_HOME}/data2 -R and **chown omm:wheel ${BIGDATA_DATA_HOME}/data2 -R**
3. On the Manager portal, choose **Cluster** > *Name of the desired cluster* > **Services** > **Yarn** > **Instance**. Select the NodeManager instance of the corresponding host, click **Instance Configuration**, and select **All Configurations**.

   Change the value of **yarn.nodemanager.local-dirs** or **yarn.nodemanager.log-dirs** to the new target directory.

For example, change the value of **yarn.nodemanager.local-dirs** or **yarn.nodemanager.log-dirs** to **/srv/BigData/data2/nm/containerlogs**.

4. Click **Save**, and then click **OK**. Restart the Yarn service.

   Click **Finish** when the system displays "Operation successful". Yarn is successfully started. No further action is required.

Step 3 Change the storage directory of a single NodeManager instance.

1. Choose **Cluster** > *Name of the desired cluster* > **Service** > **Yarn** > **Instance**, select the NodeManager instance whose storage directory needs to be modified, and choose **More** > **Stop**.
2. Log in to the NodeManager node as user **root**, and perform the following operations:
   a. Create a target directory.

      For example, to create the target directory **${BIGDATA_DATA_HOME}/data2**, run the following command:

      mkdir ${BIGDATA_DATA_HOME}/data2

   b. Mount the target directory to the new disk.

      For example, mount **${BIGDATA_DATA_HOME}/data2** to the new disk.

   c. Modify permissions on the new directory.

      For example, to modify permissions on the **${BIGDATA_DATA_HOME}/data2** directory, run the following commands:

      chmod 750 ${BIGDATA_DATA_HOME}/data2 -R and **chown omm:wheel ${BIGDATA_DATA_HOME}/data2 -R**
3. On Manager, click the specified NodeManager instance, and switch to the **Instance Configuration** page.

   Change the value of **yarn.nodemanager.local-dirs** or **yarn.nodemanager.log-dirs** to the new target directory.

   For example, change the value of **yarn.nodemanager.local-dirs** or **yarn.nodemanager.log-dirs** to **/srv/BigData/data2/nm/containerlogs**.
4. Click **Save**, and then click **OK** to restart the NodeManager instance.

   Click **Finish** when the system displays "Operation successful". The NodeManager instance is successfully started.

----End

27.8.6 Configuring YARN Big Job Scanning

YARN's big job scanning function monitors local temporary files (such as shuffle files) and key HDFS directories (OBS is not supported) for Hive, HetuEngine, and Spark jobs. It reports events when jobs consume excessive storage resources (local disks or key HDFS directories).

NOTE

This section applies only to MRS 3.5.0 and later versions.

For details about the monitored HDFS directories, see **Table 27-25**.

Table 27-25 Monitored HDFS directories

Component	Monitored HDFS directories	Threshold
Hive	hdfs://hacluster/tmp/hive-scratch/*/	400G
Hetu	hdfs://hacluster/hetuserverhistory/*/coordinator/	100G
Spark	hdfs://hacluster/sparkJobHistory/	100G

For details about the parameters, see **Table 27-26**.

Go to the **All Configurations** page of YARN and enter a parameter name in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

NOTICE

- To activate the Hive component configuration in the big job scanning feature, set **hive-ext.record.mr.applicationid** to **true**. Here are the steps you need to take:

  Go to the **All Configurations** page of the Hive service by referring to **Modifying Cluster Service Configuration Parameters**, choose **HiveServer (Role)** > **Custom**, and add **hive-ext.record.mr.applicationid** in the **hive.server.customized.configs** parameter. Set the added parameter to **true** and save the configuration.
- Currently, the Hive large job scanning feature applies only to the MapReduce engine.

Table 27-26 Parameter configuration

Parameter	Description	Default Value
hetu.job.hdfs.monitor.dir	Large directory monitoring path of HetuEngine jobs. The root directory cannot be monitored. If the directories to be monitored include variable directories such as user directories, replace them with **/*/**.	hdfs://hacluster/hetuserverhistory/*/coordinator/

Parameter	Description	Default Value
hetu.job.appId.parser.rule	Rule for extracting job IDs in the big directory monitoring path of HetuEngine jobs. Examples: • **{subdir}/{appid}**: The job ID is in the subdirectory of the monitoring directory. The subdirectory name is not fixed. • **{appid}**: The job ID is in the monitoring directory.	{appid}
hetu.job.hdfs.dir.threshold	Large directory threshold of HetuEngine jobs. If the threshold is exceeded, an event is reported. Unit: GB	100
hive.job.hdfs.monitor.dir	Big directory monitoring path of Hive jobs. The root directory cannot be monitored. If the directories to be monitored include variable directories such as user directories, replace them with **/*/**.	hdfs://hacluster/tmp/hive-scratch/*/
hive.job.appId.parser.rule	Rule for extracting job IDs in the big directory monitoring path of Hive jobs. Some examples are as follows: • **{subdir}/{appid}**: The job ID is in the subdirectory of the monitoring directory. The subdirectory name is not fixed. • **{appid}**: The job ID is in the monitoring directory.	{subdir}/{appid}
hive.job.hdfs.dir.threshold	Big directory threshold for monitoring Hive jobs. If the threshold is exceeded, an event is reported. Unit: GB	400
spark.job.hdfs.monitor.dir	Big directory monitoring path for monitoring Spark jobs. The root directory cannot be monitored. If the directories to be monitored include variable directories such as user directories, replace them with **/*/**.	hdfs://hacluster/sparkJobHistory/

Parameter	Description	Default Value
spark.job.appId.parser.rule	Rule for extracting the job ID in the large directory monitoring path of the monitored Spark job. For example: • **{subdir}/{appid}**: The job ID is in the subdirectory of the monitoring directory. The subdirectory name is not fixed. • **{appid}**: The job ID is in the monitoring directory.	{appid}
spark.job.hdfs.dir.threshold	Large directory threshold for monitoring Spark jobs. If the threshold is exceeded, an event is reported. Unit: GB	100
job.monitor.local.thread.pool	Number of threads for obtaining information about big jobs monitored by NodeManager.	50
max.job.count	Number of big jobs displayed in the event.	10
job.monitor.local.dir.threshold	Threshold of the size of the job directory on NodeManager's local disk. An event will be triggered once this threshold is reached. Unit: GB	20
job.monitor.check.period	Big job monitoring period. Setting the value to **0** disables big job monitoring. Unit: minute	10

27.9 Common Issues About Yarn

27.9.1 Why Mounted Directory for Container is Not Cleared After the Completion of the Job While Using CGroups?

Question

Why mounted directory for Container is not cleared after the completion of the job while using CGroups?

Answer

The mounted path for the Container should be cleared even if job is failed.

This happens due to the deletion timeout. Some task takes more time to complete than the deletion time.

To avoid this scenario, you can go to the **All Configurations** page of Yarn by referring to **Modifying Cluster Service Configuration Parameters**. Search for the **yarn.nodemanager.linux-container-executor.cgroups.delete-timeout-ms** configuration item in the search box to change the deletion interval. The value is in milliseconds.

27.9.2 Why the Job Fails with HDFS_DELEGATION_TOKEN Expired Exception?

Question

Why is the HDFS_DELEGATION_TOKEN expired exception reported when a job fails in security mode?

Answer

HDFS_DELEGATION_TOKEN expires because the token is not updated or it is accessed after max. lifetime.

Ensure the following parameter value of max. lifetime of the token is greater than the job running time.

dfs.namenode.delegation.token.max-lifetime=**604800000** (1 week by default)

Go to the **All Configurations** page of HDFS by referring to **Modifying Cluster Service Configuration Parameters** and search for this parameter in the search box.

NOTE

You are advised to set this parameter to a value that is multiple times of the number of hours within the max. lifecycle of the token.

27.9.3 Why Are Local Logs Not Deleted After YARN Is Restarted?

Question

If Yarn is restarted in either of the following scenarios, local logs will not be deleted as scheduled and will be retained permanently:

- When Yarn is restarted during task running, local logs are not deleted.
- When the task is complete and logs fail to be collected, restart Yarn before the logs are cleared as scheduled. In this case, local logs are not deleted.

Answer

NodeManager has a restart recovery mechanism. For details, visit the following:

Versions earlier than MRS 3.2.0: **https://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-site/NodeManager.html#NodeManager_Restart**

MRS 3.2.0 or later: **https://hadoop.apache.org/docs/r3.3.1/hadoop-yarn/hadoop-yarn-site/NodeManager.html#NodeManager_Restart**

Go to the **All Configurations** tab page of YARN by referring to **Modifying Cluster Service Configuration Parameters**. Set **yarn.nodemanager.recovery.enabled** of NodeManager to **true** to make the configuration take effect. The default value is **true**. In this way, redundant local logs are periodically deleted when the YARN is restarted.

27.9.4 Why the Task Does Not Fail Even Though AppAttempts Restarts for More Than Two Times?

Question

Why the task does not fail even though AppAttempts restarts due to failure for more than two times?

Answer

During the task execution process, if the **ContainerExitStatus** returns value **ABORTED**, **PREEMPTED**, **DISKS_FAILED**, or **KILLED_BY_RESOURCEMANAGER**, the system will not count it as a failed attempt. Therefore, the task fails only when the AppAttempts fails actually, that is, the return value is not **ABORTED**, **PREEMPTED**, **DISKS_FAILED**, or **KILLED_BY_RESOURCEMANAGER** for two times.

27.9.5 Why Is an Application Moved Back to the Original Queue After ResourceManager Restarts?

Question

After I moved an application from one queue to another, why is it moved back to the original queue after ResourceManager restarts?

Answer

This problem is caused by the constraints of the ResourceManager. If a running application is moved to another queue, information about the new queue will not be stored in the ResourceManager after the ResourceManager restarts.

Assume that a user submits a MapReduce application to the leaf queue test11. If the leaf queue test11 is deleted when the application is running, the application will go to the lost_and found queue and the application stops. To start the application, the user moves the application to the leaf queue test21 and the application resumes running. If the ResourceManager restarts, the displayed submission queue is lost_and_found, but not test21.

If the application is not complete, the ResourceManager only stores the queue information before the application is moved. As a result, the application is moved back to the original queue. To solve this problem, move the application again after the ResourceManager is restarted to write information about the new queue to the ResourceManager.

27.9.6 Why Does Yarn Not Release the Blacklist Even All Nodes Are Added to the Blacklist?

Question

Why does Yarn not release the blacklist even all nodes are added to the blacklist?

Answer

In Yarn, when the number of application nodes added to the blacklist by ApplicationMaster (AM) reaches a certain proportion (the default value is 33% of the total number of nodes), the AM automatically releases the blacklist. In this way, all available nodes are added to the blacklist and tasks can obtain node resources.

Assume that there are 8 nodes in a cluster and they are divided in to pool A and pool B by NodeLabel. There are two nodes in pool B. A user submits a task App1 to pool B, but there is not sufficient HDFS space and App1 fails to run. As a result, two nodes in pool B are added to the blacklist by the AM of App1. According to the preceding principles, 2 is less than the 33% of 8. Therefore, Yarn does not release the blacklist, and App1 cannot obtain resources and keeps running. Even if the node that is added to the blacklisted is recovered, App1 still cannot obtain resources.

The preceding principles do not apply to the resource pool scenario. Therefore, you can change the value of the client parameter (The path is *client installation path*/**Yarn/config/yarn-site.xml**) **yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold** to **(nodes number of pool / total nodes)* 33%** to solve this problem.

27.9.7 Why Does the Switchover of ResourceManager Occur Continuously?

Question

The switchover of ResourceManager occurs continuously when multiple, for example 2,000, tasks are running concurrently, causing the Yarn service unavailable.

Answer

The cause is that the time of full GabageCollection exceeds the interaction duration threshold between the ResourceManager and ZooKeeper duration threshold. As a result, the connection between the ResourceManager and ZooKeeper fails and the switchover of ResourceManager occurs continuously.

When there are multiple tasks, ResourceManager saves the authentication information about multiple tasks and transfers the information to NodeManagers through heartbeat, which is called heartbeat response. The lifecycle of heartbeat response is short. The default value is 1s. Normally, heartbeat response can be reclaimed during the JVM minor GabageCollection. However, if there are multiple tasks and there are a lot of nodes, for example 5000 nodes, in the cluster, the

heartbeat response of multiple nodes occupy a large amount of memory. As a result, the JVM cannot completely reclaim the heartbeat response during minor GabageCollection. The heartbeat response failed to be reclaimed accumulate and the JVM full GabageCollection is triggered. The JVM GabageCollection is in a blocking mode, in other words, no jobs are performed during the GabageCollection. Therefore, if the duration of full GabageCollection exceeds the periodical interaction duration threshold between the ResourceManager and ZooKeeper, the switchover occurs.

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Yarn**, and click the **Configurations** tab and then **All Configurations**. In the navigation pane on the left, choose **Yarn** > **Customization**, and add the **yarn.resourcemanager.zk-timeout-ms** parameter to the **yarn.yarn-site.customized.configs** file to increase the threshold of the periodic interaction duration between ResourceManager and ZooKeeper (the value range is less than or equal to 90,000 ms). In this way, the problem of continuous active/standby ResourceManager switchover can be solved.

27.9.8 Why Does a New Application Fail If a NodeManager Has Been in Unhealthy Status for 10 Minutes?

Question

Why does a new application fail if a NodeManager has been in unhealthy status for 10 minutes?

Answer

When **nodeSelectPolicy** is set to **SEQUENCE** and the first NodeManager connected to the ResourceManager is unavailable, the ResourceManager attempts to assign tasks to the same NodeManager in the period specified by **yarn.nm.liveness-monitor.expiry-interval-ms**.

You can use either of the following methods to avoid the preceding problem:

- Use another nodeSelectPolicy, for example, **RANDOM**.
- Go to the **All Configurations** page of Yarn by referring to **Modifying Cluster Service Configuration Parameters**. Search for the following parameters in the search box and modify the following attributes in the **yarn-site.xml** file:

  yarn.resourcemanager.am-scheduling.node-blacklisting-enabled = **true**;

  yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold = **0.5**.

27.9.9 Why Does an Error Occur When I Query the ApplicationID of a Completed or Non-existing Application Using the RESTful APIs?

Question

Why does an error occur when I query the applicationID of a completed or non-existing application using the RESTful APIs?

Answer

The Superior scheduler only stores the applicationIDs of running applications. If you view the applicationID of a completed or non-existing application by accessing the RESTful API at **https://<SS_REST_SERVER>/ws/v1/sscheduler/applications/{application_id}**. the 404 error is returned by the server. If Chrome web browser is used, the **Error Occurred** message is displayed because chrome preferentially responds in the application/xml format. If Internet Explorer is used, the **404** error code is displayed because IE web browser preferentially responds in the application/json format.

27.9.10 Why May A Single NodeManager Fault Cause MapReduce Task Failures in the Superior Scheduling Mode?

Question

In Superior scheduling mode, if a single NodeManager is faulty, why may the MapReduce tasks fail?

Answer

In normal cases, when the attempt of a single task of an application fails on a node for three consecutive times, the AppMaster of the application adds the node to the blacklist. Then, the AppMaster instructs the scheduler not to schedule the task to the node to avoid task failure.

However, by default, if 33% nodes in the cluster are added to the blacklist, the scheduler ignores the blacklisted nodes. Therefore, the blacklist feature is prone to become invalid in small cluster scenarios. For example, there are only three nodes in the cluster. If one node is faulty, the blacklist mechanism becomes invalid. The scheduler continues to schedule the task to the node no matter how many times the attempt of the task fails on the node. As a result, the number of attempts of the task reaches the maximum (4 times by default for MapReduce). And the MapReduce tasks failed.

Workaround:

The **yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold** parameter (modify the parameters in the **Yarn-site.xml** file in the *client installation path*/**Yarn/config directory**) indicates the threshold for ignoring blacklisted nodes, in percentage. You are advised to increase the value of this parameter based on the cluster scale. For example, you are advised to set this parameter to **50%** for a three-node cluster.

NOTE

The framework design of the Superior scheduler is time-based asynchronous scheduling. When the NodeManager is faulty, ResourceManager cannot quickly detect that the NodeManager is faulty (10 minutes by default). Therefore, the Superior scheduler still schedules tasks to the node, causing task failures.

27.9.11 Why Are Applications Suspended After They Are Moved From Lost_and_Found Queue to Another Queue?

Question

When a queue is deleted when there are applications running in it, these applications are moved to the "lost_and_found" queue. When these applications are moved back to another healthy queue, some tasks are suspended.

Answer

If no label expression is set for the current application, the default label expression of the queue is used as label expression for new container/resource demands requested by the application. If there is no default label expression of the queue, then **default label** is considered as the label expression for new container/resource demands requested by the application.

When application app1 is submitted to the queue Q1, **label1**, the default label expression of the queue, is used for the application's new resource requests/containers. If Q1 is deleted when app1 is running, app1 is moved to the "lost_and_found" queue. Because there is no label expression of the "lost_and_found" queue, **default label** is used as the label expression of app1's new resource requests/containers. Assume that app1 is moved to another normal queue Q2. If Q2 supports **label1** and **default label**, app1 can run properly. If Q2 does not support **label1** or **default label**, the resource request with **label1** or **default label** cannot obtain resources, causing task suspension.

To solve this problem, ensure that the queue to which the application is moved from "lost_and_found" queue supports label expression of the moved application.

You are not advised to delete a queue in which there are running applications.

27.9.12 How Do I Limit the Size of Application Diagnostic Messages Stored in the ZKstore?

Question

How do I limit the size of application diagnostic messages stored in the ZKstore?

Answer

In some cases, it has been observed that diagnostic messages may grow infinitely. Because diagnostic messages are stored in the ZKstore, it is not recommended that you allow diagnostic messages to grow indefinitely. Therefore, a property parameter is needed to set the maximum size of the diagnostic message.

If you need to set **yarn.app.attempt.diagnostics.limit.kc**, go to the **All Configurations** page by referring to **Modifying Cluster Service Configuration Parameters** and search for the following parameters in the search box:

Table 27-27 Parameter description

Parameter	Description	Default Value
yarn.app.attempt.diagnostics.limit.kc	Data size of the diagnosis message for each application connection, in kilobytes (number of characters x 1,024). When ZooKeeper is used to store the behavior status of applications, the size of diagnosis messages needs to be limited to prevent Yarn from overloading ZooKeeper. If **yarn.resourcemanager.state-store.max-completed-applications** is set to a large value, you need to decrease the value of this property to limit the total size of stored data.	64

27.9.13 Why Does a MapReduce Job Fail to Run When a Non-ViewFS File System Is Configured as ViewFS?

Question

Why does a MapReduce job fail to run when a non-ViewFS file system is configured as ViewFS?

Answer

When a non-ViewFS file system is configured as a ViewFS using cluster, the user permissions on folders in the ViewFS file system are different from those of non-ViewFS folders in the default NameService. The submitted MapReduce job fails to be executed because the directory permissions are inconsistent.

When configuring the ViewFS user in the cluster, you need to check and verify the directory permissions. Before submitting a job, change the ViewFS folder permissions based on the default NameService folder permissions.

The following table lists the default permission structure of directories configured in ViewFS. If the configured directory permissions are not included in the following table, you must change the directory permissions accordingly.

Table 27-28 Default permission structure of directories configured in ViewFS

Parameter	Description	Default Value	Default value and default permissions on the parent directory
yarn.nodemanager.remote-app-log-dir	On the default file system (usually HDFS), specify the directory to which the NM aggregates logs.	logs	777
yarn.nodemanager.remote-app-log-archive-dir	Directory for archiving logs	-	777
yarn.app.mapreduce.am.staging-dir	Staging directory used when a job is submitted	/tmp/hadoop-yarn/staging	777
mapreduce.jobhistory.intermediate-done-dir	Directory for storing historical files of MapReduce jobs	${yarn.app.mapreduce.am.staging-dir}/history/done_intermediate	777
mapreduce.jobhistory.done-dir	Directory of historical files managed by the MR JobHistory Server.	${yarn.app.mapreduce.am.staging-dir}/history/done	777

27.9.14 Why Do Reduce Tasks Fail to Run in Some OSs After the Native Task Feature is Enabled?

Question

After the Native Task feature is enabled, Reduce tasks fail to run in some OSs.

Answer

When **-Dmapreduce.job.map.output.collector.class=org.apache.hadoop.mapred.nativetask.NativeMapOutputCollectorDelegator** is executed to enable the Native Task feature during the running of MapReduce tasks that contain Reduce tasks, the tasks fail to run in some OSs, and the error message "version 'GLIBCXX_3.4.20' not found" is displayed in logs. The cause is that the GLIBCXX version of the OSs is too early. As a result, the libnativetask.so.1.0.0 library on which the feature depends cannot be loaded, leading to task failures.

Workaround:

Set **mapreduce.job.map.output.collector.class** to **org.apache.hadoop.mapred.MapTask$MapOutputBuffer**.

28 Using ZooKeeper

28.1 Using ZooKeeper from Scratch

ZooKeeper is an open-source, highly reliable, and distributed consistency coordination service. ZooKeeper is designed to solve the problem that data consistency cannot be ensured for complex and error-prone distributed systems. There is no need to develop dedicated collaborative applications, which is suitable for high availability services to ensure data consistency.

Background Information

Before using the client, you need to download and update the client configuration file on all clients except the client of the active management node.

Procedure

Step 1 Download the client configuration file.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.
   - Versions earlier than MRS 3.3.0: Choose **Cluster** > *Name of the desired cluster* > **Dashboard** > **More** > **Download Client**.
   - MRS 3.3.0 and later: In the upper right corner of the homepage, click **More** and select **Download Client**.
2. Download the cluster client.

   Set **Select Client Type** to **Configuration Files Only**, select a platform type, and click **OK** to generate the client configuration file which is then saved in the **/tmp/FusionInsight-Client/** directory on the active management node by default.

Step 2 Log in to the active management node of Manager.

1. Log in to any node where Manager is deployed as user **root**.
2. Run the following command to identify the active and standby nodes:

   sh ${BIGDATA_HOME}/om-server/om/sbin/status-oms.sh

In the command output, the value of **HAActive** for the active management node is **active**, and that for the standby management node is **standby**. In the following example, **node-master1** is the active management node, and **node-master2** is the standby management node.

```
HAMode
double
NodeName            HostName        HAVersion       StartTime           HAActive
HAAllResOK          HARunPhase
192-168-0-30        node-master1    V100R001C01     2020-05-01 23:43:02 active
normal              Actived
192-168-0-24        node-master2    V100R001C01     2020-05-01 07:14:02 standby
normal              Deactived
```

3. Log in to the primary management node as user **root** and run the following command to switch to user **omm**:

   sudo su - omm

Step 3 Run the following command to switch to the client installation directory, for example, **/opt/client**:

cd /opt/client

Step 4 Run the following command to update the client configuration for the active management node.

sh refreshConfig.sh /opt/client *Full path of the client configuration file package*

For example, run the following command:

sh refreshConfig.sh /opt/client /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Services_Client.tar

If the following information is displayed, the configurations have been updated successfully:

```
ReFresh components client config is complete.
Succeed to refresh components client config.
```

Step 5 Use the client on a Master node.

1. On the active management node where the client is updated, for example, node **192-168-0-30**, run the following command to go to the client directory:

   cd /opt/client

2. Run the following command to configure environment variables:

   source bigdata_env

3. If Kerberos authentication has been enabled for the current cluster, run the following command to authenticate the current user. For details, see **Managing Roles** to configure roles with required permissions. For details about how to bind roles with users, see **Creating a User**. If Kerberos authentication is disabled for the current cluster, skip this step:

   kinit *MRS cluster user*

   Example: **kinit zookeeperuser**.

4. Run the following Zookeeper client command:

   zkCli.sh -server *<zookeeper installation node IP>:<port>*

   Example: **zkCli.sh -server node-master1DGhZ:2181**

NOTE

You can search for **clientPort** in all ZooKeeper configuration parameters to obtain the value of ***<port>***. The default ports are as follows:

- The default open-source port number is **2181**.
- The default customized port number is **24002**.

Port customization/open source: When creating an LTS version cluster, you can set **Component Port** to **Open source** or **Custom**. If **Open source** is selected, the open source port is used. If **Custom** is selected, the customized port is used.

Step 6 Run the ZooKeeper client command.

1. Create a ZNode.
   ```
   create /test
   ```
2. View ZNode information.
   ```
   ls /
   ```
3. Write data to the ZNode.
   ```
   set /test "zookeeper test"
   ```
4. View the data written to the ZNode.
   ```
   get /test
   ```
5. Delete the created ZNode.
   ```
   delete /test
   ```

----End

28.2 Configuring the ZooKeeper Permissions

Scenario

Configure znode permission of ZooKeeper.

ZooKeeper uses an access control list (ACL) to implement znode access control. The ZooKeeper client specifies a znode ACL, and the ZooKeeper server determines whether a client that requests for a znode has related operation permission according to the ACL. ACL configuration involves the following four operations:

- Check znode ACLs in ZooKeeper.
- Add znode ACLs to ZooKeeper.
- Modify znode ACLs in ZooKeeper.
- Delete znode ACLs from ZooKeeper.

  The ZooKeeper ACL permission is described as follows:

  ZooKeeper supports five types of permission, create, delete, read, write, and admin. ZooKeeper permission control is of a znode level. That is, the permission configuration for a parent znode is not inherited by its child znodes. The ZooKeeper znode default permission is **world:anyone: cdrwa**. That is, any user has all permissions.

NOTE

ACL has three parts:

The first part is the authentication type. For example, **world** indicates all authentication types and **sasl** indicates the kerberos authentication type.

The second part is the account. For example, anyone indicates any user.

The third part is permission. For example, **cdrwa** indicates all permissions.

In particular, because starting the client in common mode does not need authentication, ACL with **sasl** authentication type cannot be used in common mode. Authentications of **sasl** scheme in this document are performed in clusters that have the security mode enabled.

Table 28-1 Five types of ZooKeeper ACLs

Permission Description	Permission Name	Permission Details
Create permission	create(c)	Users with this permission can create child znodes in the current znode.
Delete permission	delete(d)	Users with this permission can delete the current znode.
Read permission	read(r)	Users with this permission can obtain data of the current znode and list all the child znodes of the current znode.
Write permission	write(w)	Users with this permission can write data to the current znode and its child znodes.
Administration permission	admin(a)	Users with this permission can set permission for the current znode.

Impact on the System

NOTICE

Modifying ZooKeeper ACLs is a critical operation. If znode permission is modified in ZooKeeper, other users may have no permission to access the znode and some system functions are abnormal. In 3.5.6 and later versions, users must have the read permission for the **getAcl** operation.

Prerequisites

- The ZooKeeper client has been installed in a directory, for example, **/opt/client**.
- You have obtained the username and password of an MRS cluster administrator.

Procedure

Start the ZooKeeper client.

Step 1 Log in to the server where the ZooKeeper client is installed as user **root**.

Step 2 Run the following command to go to the client installation directory:

cd /opt/client

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 Run the following command and enter the user password to authenticate the user's identity (This step is required only for clusters in security mode, and user **userA** is provided as an example of an authorized user.):

kinit userA

Step 5 On the ZooKeeper client, run the following command to go to the ZooKeeper command-line interface (CLI):

sh zkCli.sh -server *ZooKeeper plane IP address of any instance*:*clientPort*

The default **clientPort** is **2181**.

Example: **sh zkCli.sh -server 192.168.0.151:2181**

Step 6 Run the **ls** command to view the znode list in ZooKeeper. For example, you can view the list of znodes in the root directory.

ls /

```
[zk: 192.168.0.151:2181(CONNECTED) 1] ls /
[hadoop-flag, hadoop-ha, test, test2, test3, test4, test5, test6, zookeeper]
```

View the ZooKeeper znode ACL.

Step 7 Start the ZooKeeper client.

Step 8 Run the **getAcl** command to view znodes. The following command can be used to view the created znode ACL named **test**:

getAcl */znode name*

```
[zk: 192.168.0.151:2181(CONNECTED) 2] getAcl /test
'world,'anyone
: cdrwa
```

Add a ZooKeeper znode ACL.

Step 9 Start the ZooKeeper client.

Step 10 View the old ACL information to check whether the current account has the permission to modify the znode ACL information (a permission). If no, use kinit to switch to a user that has the permission and restart the ZooKeeper client.

getAcl */znode name*

```
 [zk: 192.168.0.151:2181(CONNECTED) 3] getAcl /test
'world,'anyone
: cdrwa
```


Step 11 Run the **setAcl** command to add an ACL. The command for adding an ACL is as follows:

setAcl /test world:anyone:cdrwa,sasl: *username*@: *<system domain name>:ACL value*

For example, to create the ACL of user **admin** to the test znode, run the following command:

setAcl /test world:anyone:cdrwa,sasl:userA@HADOOP.COM:cdrwa

NOTE

When adding a new ACL, reserve the existing ones. The new and old ACLs are separated by a comma. The newly added ACL has three parts:

- The first part is the authentication type. For example, **sasl** indicates kerberos authentication.
- The second part is the account. For example, **userA@HADOOP.COM** indicates user **userA**.
- The third part is permission. For example, **cdrwa** indicates all permissions.

Step 12 After adding the ACL, run the **getAcl** command to check whether the permission is added successfully:

getAcl */znode name*

```
[zk: 192.168.0.151:2181(CONNECTED) 4] getAcl /test
'world,'anyone
: cdrwa
'sasl,'userA@<System domain name>
: cdrwa
```

Modify the ZooKeeper znode ACL.

Step 13 Start the ZooKeeper client.

Step 14 View the old ACL information to check whether the current account has the permission to modify the znode ACL information (a permission). If no, use kinit to switch to a user that has the permission and restart the ZooKeeper client.

getAcl */znode name*

```
[zk: 192.168.0.151:2181(CONNECTED) 5] getAcl /test
'world,'anyone
: cdrwa
'sasl,'userA@<System domain name>
: cdrwa
```

Step 15 Run the **setAcl** command to modify an ACL. The command for adding an ACL is as follows:

s**etAcl /test sasl:***Username@<System domain name>:ACL value*

For example, to reserve all permissions of user **userA** and delete the rw permission of user **anyone**, run the following command:

setAcl /test sasl:userA@HADOOP.COM:cdrwa

Step 16 After modifying the ACL, run the **getAcl** command to check whether the permission is modified successfully:

getAcl */znode name*

```
[zk: 192.168.0.151:2181(CONNECTED) 6] getAcl /test
'sasl,'userA@<System domain name>
: cdrwa
```

Delete the ZooKeeper znode ACL.

Step 17 Start the ZooKeeper client.

Step 18 View the old ACL information to check whether the current account has the permission to modify the znode ACL information (a permission). If no, use kinit to switch to a user that has the permission and restart the ZooKeeper client.

getAcl */znode name*

```
[zk: 192.168.0.151:2181(CONNECTED) 5] getAcl /test
'world,'anyone
: rw
'sasl,'userA@<System domain name>
: cdrwa
```

Step 19 Run the **setAcl** command to add an ACL. The command for adding an ACL is as follows:

setAcl /test sasl:*Username@<System domain name>:ACL value*

For example, to reserve all permissions of user **userA** and delete the rw permission of user **anyone**, run the following command:

setAcl /test sasl:userA@HADOOP.COM:cdrwa

Step 20 After modifying the ACL, run the **getAcl** command to check whether the permission is modified successfully:

getAcl */znode name*

```
[zk: 192.168.0.151:2181(CONNECTED) 6] getAcl /test
'sasl,'userA@<System domain name>
: cdrwa
```

----End

28.3 ZooKeeper Common Configuration Parameters

Navigation path for setting parameters:

Go to the **All Configurations** page of ZooKeeper by referring to **Modifying Cluster Service Configuration Parameters**. Enter a parameter name in the search box.

Table 28-2 Parameters

Parameter	Description	Default Value
skipACL	Specifies whether to skip the permission check of the ZooKeeper node.	no

Parameter	Description	Default Value
maxClientCnxns	Specifies the maximum number of connections of ZooKeeper. It is recommended this parameter is set to a larger value in scenarios with a large number of connections.	2000
LOG_LEVEL	Specifies the log level. This parameter can be set to **DEBUG** during commissioning.	INFO
acl.compare.shortName	Specifies whether to perform ACL authentication only by principal username when the Znode ACL authentication type is SASL.	true
synclimit	Specifies the interval of synchronization between the follower and leader (unit: tick). If the leader does not respond within the specified time range, the connection cannot be established.	15
tickTime	Specifies the duration of a tick (in milliseconds). It is the basic time unit used by ZooKeeper, which defines heartbeat and timeout durations.	4000

NOTE

The ZooKeeper internal time is determined by **ticktime** and **synclimit**. To increase the ZooKeeper internal timeout interval, increase the timeout interval for the client to connect to ZooKeeper.

28.4 ZooKeeper Log Overview

Log Description

Log path: **/var/log/Bigdata/zookeeper/quorumpeer** (Run log), **/var/log/Bigdata/audit/zookeeper/quorumpeer** (Audit log)

Log archive rule: The automatic ZooKeeper log compression function is enabled. By default, when the size of logs exceeds 30 MB, logs are automatically compressed into a log file. A maximum of 20 compressed files can be reserved. The number of compressed files can be configured on Manager.

Table 28-3 ZooKeeper log list

Log Type	Log File Name	Description
Run logs	zookeeper-<SSH_USER>-<process_name>-<hostname>.log	ZooKeeper system log file, which records most of the logs generated when the ZooKeeper system is running.
	check-serviceDetail.log	Log that records whether the ZooKeeper service starts successfully.
	zookeeper-<SSH_USER>-<DATA>-<PID>-gc.log	ZooKeeper garbage collection log file
	instanceHealthDetail.log	Log that records the health check details of ZooKeeper instance
	zookeeper-omm-server-<hostname>.out	Log indicating that ZooKeeper unexpectedly quits
	zk-err-<zkpid>.log	ZooKeeper fatal error log
	java_pid<zkpid>.hprof	ZooKeeper memory overflow log
	funcDetail.log	ZooKeeper instance startup log
	zookeeper-period-check.log	Health check log of the ZooKeeper instance
	zookeeper-period-check-java.log	ZooKeeper quota monitoring period check log
Audit Log	zk-audit-quorumpeer.log	ZooKeeper operation audit log

Log levels

Table 28-4 describes the log levels supported by ZooKeeper. The priorities of log levels are FATAL, ERROR, WARN, INFO, and DEBUG in descending order. Logs whose levels are higher than or equal to the specified level are printed. The number of printed logs decreases as the specified log level increases.

Table 28-4 Log levels

Level	Description
FATAL	Logs of this level record fatal error information about the current event processing that may result in a system crash.
ERROR	Error information about the current event processing, which indicates that system running is abnormal.
WARN	Abnormal information about the current event processing. These abnormalities will not result in system faults.
INFO	Logs of this level record normal running status information about the system and events.
DEBUG	Logs of this level record the system information and system debugging information.

To modify log levels, perform the following operations:

Step 1 Go to the **All Configurations** page of the ZooKeeper service by referring to **Modifying Cluster Service Configuration Parameters**.

Step 2 On the menu bar on the left, select the log menu of the target role.

Step 3 Select a desired log level.

Step 4 Click **Save**. In the displayed dialog box, click **OK** to make the configuration take effect.

NOTE

The configurations take effect immediately without the need to restart the service.

----End

Log Format

The following table lists the ZooKeeper log formats.

Table 28-5 Log Format

Log Type	Component	Format	Example
Run logs	zookeeper quorumpeer	<*yyyy-MM-dd HH:mm:ss,SSS*>|<*Log level*>|<*Name of the thread that generates the log*>|<*Message in the log*>|<*Location where the log event occurs*>	2020-01-20 16:33:43,816 | INFO | main | Defaulting to majority quorums | org.apache.zookeeper.server.quorum.QuorumPeerConfig.parseProperties(QuorumPeerConfig.java:335)
Audit logs	zookeeper quorumpeer	<*yyyy-MM-dd HH:mm:ss,SSS*>|<*Log level*>|<*Name of the thread that generates the log*>|<*Message in the log*>|<*Location where the log event occurs*>	2020-01-20 16:33:54,313 | INFO | CommitProcessor:13 | session=0xd4b0679daea0000 ip=10.177.112.145 operation=create znode target=ZooKeeperServer znode=/zk-write-test-2 result=success | org.apache.zookeeper.ZKAuditLogger$LogLevel$5.printLog(ZKAuditLogger.java:70)

28.5 Common Issues About ZooKeeper

28.5.1 Why Do ZooKeeper Servers Fail to Start After Many znodes Are Created?

Question

After a large number of znodes are created, ZooKeeper servers in the ZooKeeper cluster become faulty and cannot be automatically recovered or restarted.

Logs of followers:

```
2016-06-23 08:00:18,763 | WARN  | QuorumPeer[myid=26](plain=/10.16.9.138:2181)(secure=disabled) | Exception when following the leader |
```



```
org.apache.zookeeper.server.quorum.Follower.followLeader(Follower.java:93)
java.net.SocketTimeoutException: Read timed out
    at java.net.SocketInputStream.socketRead0(Native Method)
    at java.net.SocketInputStream.socketRead(SocketInputStream.java:116)
    at java.net.SocketInputStream.read(SocketInputStream.java:170)
    at java.net.SocketInputStream.read(SocketInputStream.java:141)
    at java.io.BufferedInputStream.fill(BufferedInputStream.java:246)
    at java.io.BufferedInputStream.read(BufferedInputStream.java:265)
    at java.io.DataInputStream.readInt(DataInputStream.java:387)
    at org.apache.jute.BinaryInputArchive.readInt(BinaryInputArchive.java:63)
    at org.apache.zookeeper.server.quorum.QuorumPacket.deserialize(QuorumPacket.java:83)
    at org.apache.jute.BinaryInputArchive.readRecord(BinaryInputArchive.java:99)
    at org.apache.zookeeper.server.quorum.Learner.readPacket(Learner.java:156)
    at org.apache.zookeeper.server.quorum.Learner.registerWithLeader(Learner.java:276)
    at org.apache.zookeeper.server.quorum.Follower.followLeader(Follower.java:75)
    at org.apache.zookeeper.server.quorum.QuorumPeer.run(QuorumPeer.java:1094)
2016-06-23 08:00:18,764 | INFO  | QuorumPeer[myid=26](plain=/10.16.9.138:2181)(secure=disabled) |
shutdown called | org.apache.zookeeper.server.quorum.Follower.shutdown(Follower.java:198)
java.lang.Exception: shutdown Follower
    at org.apache.zookeeper.server.quorum.Follower.shutdown(Follower.java:198)
    at org.apache.zookeeper.server.quorum.QuorumPeer.stopFollower(QuorumPeer.java:1141)
    at org.apache.zookeeper.server.quorum.QuorumPeer.run(QuorumPeer.java:1098)
```

Logs of the leader:

```
2016-06-23 07:30:57,481 | WARN  | QuorumPeer[myid=25](plain=/10.16.9.136:2181)(secure=disabled) |
Unexpected exception | org.apache.zookeeper.server.quorum.QuorumPeer.run(QuorumPeer.java:1108)
java.lang.InterruptedException: Timeout while waiting for epoch to be acked by quorum
    at org.apache.zookeeper.server.quorum.Leader.waitForEpochAck(Leader.java:1221)
    at org.apache.zookeeper.server.quorum.Leader.lead(Leader.java:487)
    at org.apache.zookeeper.server.quorum.QuorumPeer.run(QuorumPeer.java:1105)
2016-06-23 07:30:57,482 | INFO  | QuorumPeer[myid=25](plain=/10.16.9.136:2181)(secure=disabled) |
Shutdown called | org.apache.zookeeper.server.quorum.Leader.shutdown(Leader.java:623)
java.lang.Exception: shutdown Leader! reason: Forcing shutdown
    at org.apache.zookeeper.server.quorum.Leader.shutdown(Leader.java:623)
    at org.apache.zookeeper.server.quorum.QuorumPeer.stopLeader(QuorumPeer.java:1149)
    at org.apache.zookeeper.server.quorum.QuorumPeer.run(QuorumPeer.java:1110)
```

Answer

After a large number of znodes are created, a large volume of data needs to be synchronized between the follower and leader. If the data synchronization is not complete within the specified time, all ZooKeeper servers fail to start.

Go to the **All Configurations** page of the ZooKeeper service by referring to **Modifying Cluster Service Configuration Parameters**. To recover ZooKeeper servers, increase the values of **syncLimit** and **initLimit** in the ZooKeeper configuration file **zoo.cfg** until ZooKeeper servers are successfully started.

Table 28-6 Parameters

Parameter	Description	Default Value
syncLimit	Interval (unit: tick) at which data is synchronized between the follower and the leader. If the leader does not respond to the follower within the specified time, the connection between the leader and follower cannot be set up.	15

Parameter	Description	Default Value
initLimit	Interval (unit: tick) within which the connection and synchronization between the follower and leader must be completed.	15

If ZooKeeper servers do not recover even after **initLimit** and **syncLimit** are set to **300** ticks, check that no other application is killing the ZooKeeper. For example, if the parameter value is **300** and the ticket duration is 2000 ms, the maximum synchronization duration is 600s (300 x 2000 ms).

There may exist the situation where an overwhelming amount of data is created in ZooKeeper and it takes long to synchronize data between the follower and the leader and to save data to the hard disk. This means that ZooKeeper needs to run for a long time. Ensure that no other monitoring application kills the ZooKeeper while ZooKeeper is running.

28.5.2 Why Does the ZooKeeper Server Display the java.io.IOException: Len Error Log?

Question

After a large number of znodes are created in a parent directory, the ZooKeeper client will fail to fetch all child nodes of this parent directory in a single request.

Logs of client:

```
2017-07-11 13:17:19,610 [myid:] - WARN  [New I/O worker #3:ClientCnxnSocketNetty
$ZKClientHandler@468] - Exception caught: [id: 0xb66cbb85, /10.18.97.97:49192 :>
10.18.97.97/10.18.97.97:2181] EXCEPTION: java.nio.channels.ClosedChannelException
java.nio.channels.ClosedChannelException
at org.jboss.netty.handler.ssl.SslHandler$6.run(SslHandler.java:1580)
at org.jboss.netty.channel.socket.ChannelRunnableWrapper.run(ChannelRunnableWrapper.java:40)
at org.jboss.netty.channel.socket.nio.AbstractNioWorker.executeInIoThread(AbstractNioWorker.java:71)
at org.jboss.netty.channel.socket.nio.NioWorker.executeInIoThread(NioWorker.java:36)
at org.jboss.netty.channel.socket.nio.AbstractNioWorker.executeInIoThread(AbstractNioWorker.java:57)
at org.jboss.netty.channel.socket.nio.NioWorker.executeInIoThread(NioWorker.java:36)
at org.jboss.netty.channel.socket.nio.AbstractNioChannelSink.execute(AbstractNioChannelSink.java:34)
at org.jboss.netty.handler.ssl.SslHandler.channelClosed(SslHandler.java:1566)
at org.jboss.netty.channel.Channels.fireChannelClosed(Channels.java:468
at org.jboss.netty.channel.socket.nio.AbstractNioWorker.close(AbstractNioWorker.java:376)
at org.jboss.netty.channel.socket.nio.NioWorker.read(NioWorker.java:93)
at org.jboss.netty.channel.socket.nio.AbstractNioWorker.process(AbstractNioWorker.java:109)
at org.jboss.netty.channel.socket.nio.AbstractNioSelector.run(AbstractNioSelector.java:312)
at org.jboss.netty.channel.socket.nio.AbstractNioWorker.run(AbstractNioWorker.java:90)
at org.jboss.netty.channel.socket.nio.NioWorker.run(NioWorker.java:178)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
```

Logs of leader:

```
2017-07-11 13:17:33,043 [myid:1] - WARN  [New I/O worker #7:NettyServerCnxn@445] - Closing
connection to /10.18.101.110:39856
java.io.IOException: Len error 45
at org.apache.zookeeper.server.NettyServerCnxn.receiveMessage(NettyServerCnxn.java:438)
at org.apache.zookeeper.server.NettyServerCnxnFactory
$CnxnChannelHandler.processMessage(NettyServerCnxnFactory.java:267)
```



```
at org.apache.zookeeper.server.NettyServerCnxnFactory
$CnxnChannelHandler.messageReceived(NettyServerCnxnFactory.java:187)
at org.jboss.netty.channel.SimpleChannelHandler.handleUpstream(SimpleChannelHandler.java:88)
at org.jboss.netty.channel.DefaultChannelPipeline.sendUpstream(DefaultChannelPipeline.java:564)
at org.jboss.netty.channel.DefaultChannelPipeline.sendUpstream(DefaultChannelPipeline.java:559)
at org.jboss.netty.channel.Channels.fireMessageReceived(Channels.java:268)
at org.jboss.netty.channel.Channels.fireMessageReceived(Channels.java:255)
at org.jboss.netty.channel.socket.nio.NioWorker.read(NioWorker.java:88)
at org.jboss.netty.channel.socket.nio.AbstractNioWorker.process(AbstractNioWorker.java:109)
at org.jboss.netty.channel.socket.nio.AbstractNioSelector.run(AbstractNioSelector.java:312)
at org.jboss.netty.channel.socket.nio.AbstractNioWorker.run(AbstractNioWorker.java:90)
at org.jboss.netty.channel.socket.nio.NioWorker.run(NioWorker.java:178)
at org.jboss.netty.util.ThreadRenamingRunnable.run(ThreadRenamingRunnable.java:108)
at org.jboss.netty.util.internal.DeadLockProofWorker$1.run(DeadLockProofWorker.java:42)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
```

Answer

After a large number of znodes are created in a single parent directory and the client tries to fetch all the child znodes in a single request, the server will fail to return because the results exceed the data size that can be stored in a znode.

To avoid this problem, set **jute.maxbuffer** to a larger value based on the client application.

jute.maxbuffer can only be set to a Java system property without the Zookeeper prefix. To set **jute.maxbuffer** to *X*, set **Djute.maxbuffer** to *X* when starting the ZooKeeper client or the service.

For example, set the parameter to 4 MB: **-Djute.maxbuffer=0x400000**.

Table 28-7 Parameters

Parameter	Description	Default Value
jute.maxbuffer	Specifies the maximum length of data that can be stored in znode. The unit is byte. Default value: 0xfffff, which is less than 1 MB. **NOTE** If this option is changed, the system property must be set on all servers and clients, otherwise problems will arise.	0xfffff

28.5.3 Why Four Letter Commands Don't Work With Linux netcat Command When Secure Netty Configurations Are Enabled at Zookeeper Server?

Question

Why four letter commands do not work with linux netcat command when secure netty configurations are enabled at Zookeeper server?

For example,

echo stat /netcat host port

Answer

Linux ***netcat*** command does not have option to communicate Zookeeper server securely, so it cannot support Zookeeper four letter commands when secure netty configurations are enabled.

To avoid this problem, user can use below Java API to execute four letter commands.

```
org.apache.zookeeper.client.FourLetterWordMain
```

For example,

```
String[] args = new String[]{host, port, "stat"};
org.apache.zookeeper.client.FourLetterWordMain.main(args);
```

NOTE

netcat command should be used only with non secure netty configuration.

28.5.4 How Do I Check Which ZooKeeper Instance Is a Leader?

Question

How to check whether the role of a ZooKeeper instance is a Leader or Follower.

Answer

1. Log in to Manager and choose **Cluster** > **Services** > **ZooKeeper** > **Instances**.
2. On the displayed page, click the name of the quorumpeer instance.
3. On the displayed instance details page, view the **Server State** of the instance.

28.5.5 Why Cannot the Client Connect to ZooKeeper using the IBM JDK?

Question

When the IBM JDK is used, the client fails to connect to ZooKeeper.

Answer

The possible cause is that the **jaas.conf** file format of the IBM JDK is different from that of the common JDK.

If IBM JDK is used, use the following **jaas.conf** template. The **useKeytab** file path must start with **file://**, followed by an absolute path.

```
Client {
com.ibm.security.auth.module.Krb5LoginModule required
useKeytab="file://D:/install/HbaseClientSample/conf/user.keytab"
principal="hbaseuser1"
credsType="both";
};
```


28.5.6 What Should I Do When the ZooKeeper Client Fails to Refresh a TGT?

Question

The ZooKeeper client fails to refresh a TGT and therefore ZooKeeper cannot be accessed. The error message is as follows:

```
Login: Could not renew TGT due to problem running shell command: '***/kinit -R'; exception was:org.apache.zookeeper.Shell$ExitCodeException: kinit: Ticket expired while renewing credentials
```

Answer

ZooKeeper uses the system command **kinit – R** to refresh a ticket. In the current version of MRS, the function of this command is canceled. If a long-term task needs to be executed, you are advised to implement the authentication function in keytab mode.

In the *client installation path*/**ZooKeeper/zookeeper/conf/jaas.conf** configuration file, set **useTicketCache** to **false**, **useKeyTab** to **true**, and specify the keytab path.

28.5.7 Why Is Message "Node does not exist" Displayed when A Large Number of Znodes Are Deleted Using the deleteall Command

Question

When the client connects to a non-leader instance, run the **deleteall** command to delete a large number of znodes, the error message "Node does not exist" is displayed, but run the **stat** command, the node status can be obtained.

Answer

The leader and follower data is not synchronized due to network problems or large data volume.

To solve this problem, connect the client to the leader instance and delete the instance.

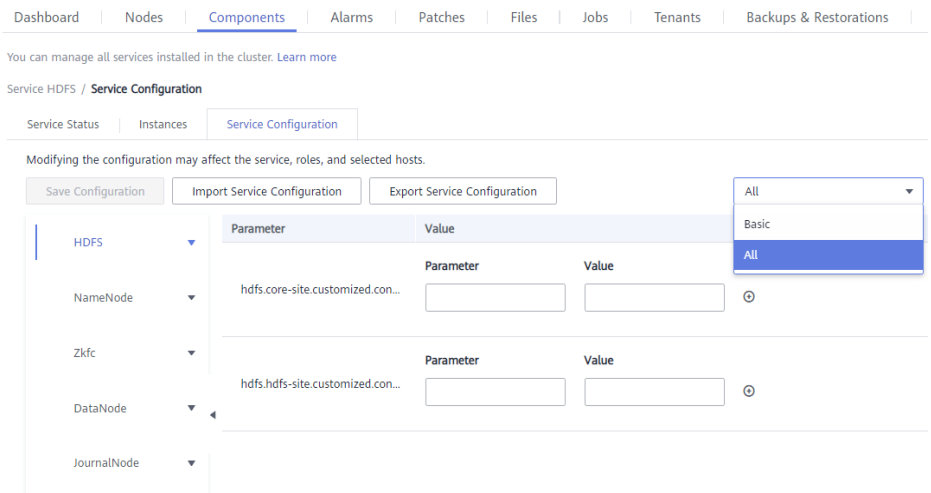
To delete the leader node, view the IP address of the node where the leader resides by referring to **How Do I Check Which ZooKeeper Instance Is a Leader?**, run the **zkCli.sh -server leader** *node IP address* **2181** command to connect to the client, and then run the **deleteall** command to delete the leader node. For details, see **Using ZooKeeper from Scratch**.

29 Appendix

29.1 Modifying Cluster Service Configuration Parameters

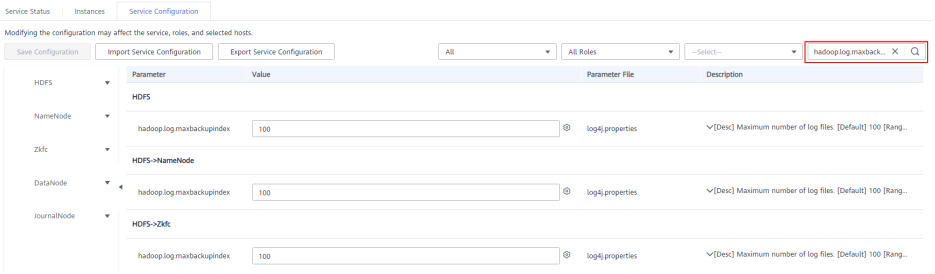
- You can modify service configuration parameters on the cluster management page of the MRS management console .
  a. Log in to the MRS console. In the left navigation pane, choose **Clusters** > **Active Clusters**, and click a cluster name.
  b. Choose **Components** > *Name of the desired service* > **Service Configuration**.

     The **Basic Configuration** tab page is displayed by default. To modify more parameters, click the **All Configurations** tab. The navigation tree displays all configuration parameters of the service. The level-1 nodes in the navigation tree are service names or role names. The parameter category is displayed after the level-1 node is expanded. (The following figure uses the HDFS component as an example.)



  c. In the navigation tree, select the specified parameter category and change the parameter values on the right.

If you are not sure about the location of a parameter, you can enter the parameter name in search box in the upper right corner. The system searches for the parameter in real time and displays the result. (The following figure uses the HDFS component as an example.)



d. Click **Save Configuration**. In the displayed dialog box, click **OK**.

e. Wait until the message **Operation successful** is displayed. Click **Finish**.

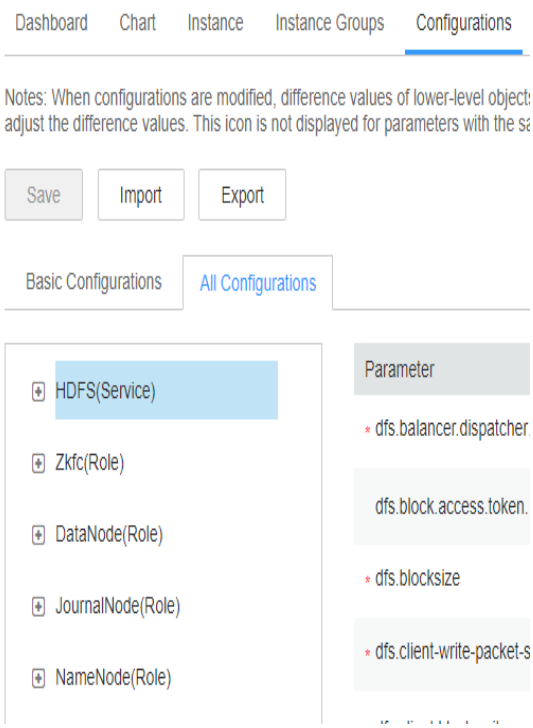
The configuration is modified.

Check whether there is any service whose configuration has expired in the cluster. If yes, restart the corresponding service or role instance for the configuration to take effect. You can also select **Restart the affected services or instances** when saving the configuration. .

You can log in to FusionInsight Manager to modify service configuration parameters.

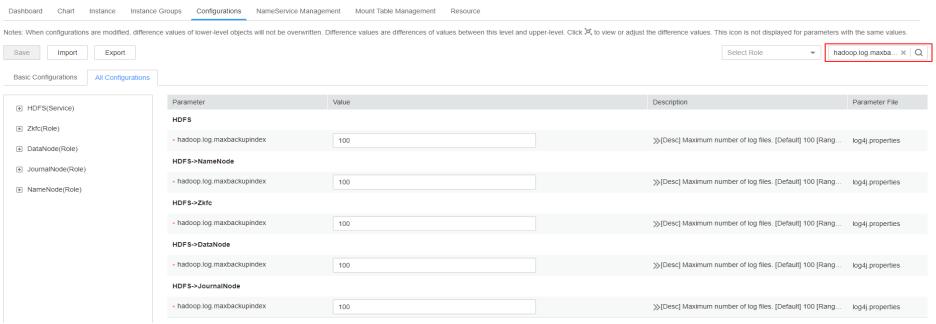
1. You have logged in to FusionInsight Manager.
2. Choose **Cluster** > **Service**.
3. Click the specified service name on the service management page.
4. Click **Configuration**.

   The **Basic Configuration** tab page is displayed by default. To modify more parameters, click the **All Configurations** tab. The navigation tree displays all configuration parameters of the service. The level-1 nodes in the navigation tree are service names or role names. The parameter category is displayed after the level-1 node is expanded. (The following figure uses the HDFS component as an example.)



5. In the navigation tree, select the specified parameter category and change the parameter values on the right.

   If you are not sure about the location of a parameter, you can enter the parameter name in search box in the upper right corner. The system searches for the parameter in real time and displays the result. (The following figure uses the HDFS component as an example.)



6. Click **Save**. In the confirmation dialog box, click **OK**.
7. Wait until the message **Operation successful** is displayed. Click **Finish**.

   The configuration is modified.

   Check whether there is any service whose configuration has expired in the cluster. If yes, restart the corresponding service or role instance for the configuration to take effect.

29.2 Accessing FusionInsight Manager

Scenario

FusionInsight Manager is used to monitor, configure, and manage clusters. After the cluster is installed, you can use the account to log in to FusionInsight Manager.

NOTE

If you cannot log in to the WebUI of the component, access FusionInsight Manager by referring to **Accessing FusionInsight Manager from an ECS**.

Accessing FusionInsight Manager Using EIP

If the EIP address function is enabled for the cluster, perform the following steps:

Step 1 Log in to the MRS management console.

Step 2 In the navigation pane, choose **Clusters** > **Active Clusters**. Click the target cluster name to access the cluster details page.

Step 3 Click **Manager** next to **MRS Manager**. In the displayed dialog box, configure the EIP information.

1. If no EIP is bound during MRS cluster creation, select an available EIP from the drop-down list on the right of **IEP**. If you have bound an EIP when you create a cluster, go to **Step 3.2**.

   NOTE

   - If no EIPs are available, click **Manage EIP** to buy one. Then, select the EIP from the drop-down list.
   - To unbind or release an EIP after using it, log in to the **EIPs** page, locate the row containing the target EIP, and click **Unbind** or choose **More** > **Release** in the **Operation** column.
   - If an EIP has been created but cannot be found during binding, the EIP may have been bound to another cluster. In this case, unbind the EIP on the **EIPs** page and then bind it to the current cluster.

2. In **Security Group**, select the security group to which the current cluster belongs. The security group is configured during cluster creation or is automatically created by the cluster.

   NOTE

   - When creating a custom cluster, you can configure a security group created in advance or retain the default value **Auto create**. When you quickly create a cluster, the security group is automatically created by the cluster.
   - You can view the security group name in **Security Group** on the **Dashboard** tab page of the cluster.

3. Add a security group rule. By default, the filled-in rule is used to access the EIP. To enable multiple IP address segments to access Manager, see steps **Step 6** to **Step 9**. If you want to view, modify, or delete a security group rule, click **Manage Security Group Rule**.

4. Select the information to be confirmed and click **OK**.

Step 4 Click **OK**. The Manager login page is displayed.

Step 5 Enter the default username **admin** and the password set during cluster creation, and click **Log In**. The Manager page is displayed.

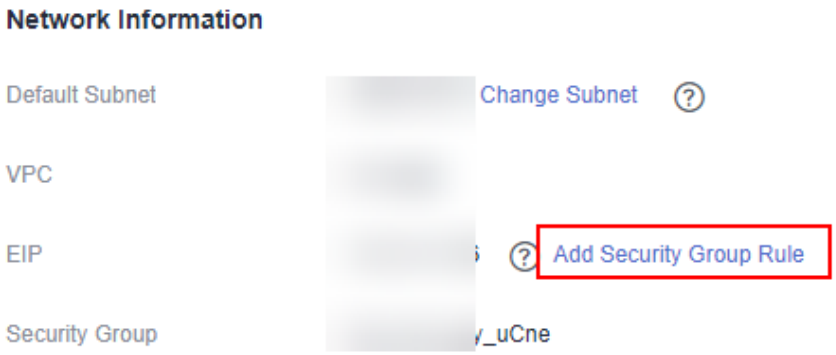
Step 6 On the MRS management console, choose **Clusters** > **Active Clusters**. Click the target cluster name to access the cluster details page.

NOTE

To grant other users the permission to access Manager, perform **Step 6** to **Step 9** to add the users' public IP addresses to the trusted IP address range.

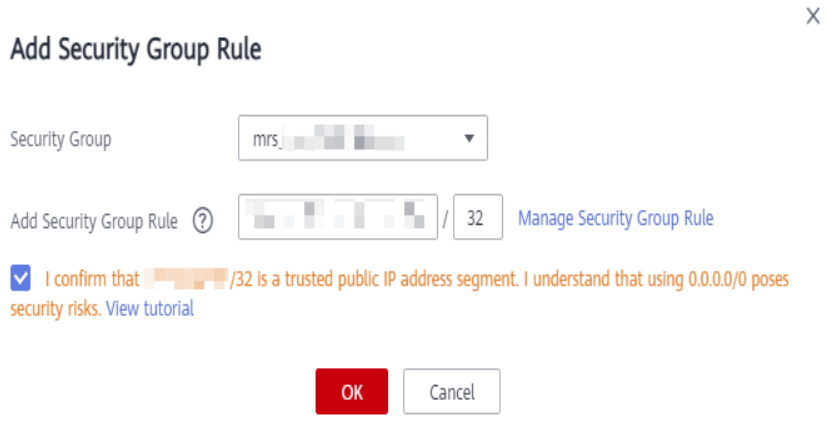
Step 7 Click **Add Security Group Rule** next to **EIP**.

Figure 29-1 Cluster details



Step 8 On the **Add Security Group Rule** page, add the IP address segment for users to access the public network and select **I confirm that *public network IP*/*port* is a trusted public IP address. I understand that using 0.0.0.0/0. poses security risks**. See **Figure 29-2**.

Figure 29-2 Adding a security group rule



By default, the IP address used for accessing the public network is filled. You can change the IP address segment as required. To enable multiple IP address

segments, repeat steps **Step 6** to **Step 9**. If you want to view, modify, or delete a security group rule, click **Manage Security Group Rule**.

Step 9 Click **OK**.

----End

Accessing FusionInsight Manager from an ECS

Step 1 On the MRS management console, click **Clusters**.

Step 2 On the **Active Clusters** page, click the name of the specified cluster.

Record the **AZ**, **VPC**, **MRS ManagerSecurity Group** of the cluster.

Step 3 On the homepage of the management console, choose **Service List** > **Elastic Cloud Server** to switch to the ECS management console and create an ECS.

- The **AZ**, **VPC**, and **Security Group** of the ECS must be the same as those of the cluster to be accessed.
- Select a Windows public image. For example, a standard image **Windows Server 2012 R2 Standard 64bit(40GB)**.
- For details about other configuration parameters, see **Purchasing an ECS with Customized Configurations**.

NOTE

If the security group of the ECS is different from **Default Security Group** of the Master node, you can modify the configuration using either of the following methods:

- Change the security group of the ECS to the default security group of the Master node. For details, see **Changing a Security Group**.
- Add two security group rules to the security groups of the Master and Core nodes to enable the ECS to access the cluster. Set **Protocol** to **TCP**, **Ports** of the two security group rules to **28443** and **20009**, respectively. For details, see **Creating a Security Group**.

Step 4 On the EIP management console, apply for an EIP and bind it to the ECS.

Step 5 Log in to the ECS.

The Windows system account, password, EIP, and the security group rules are required for logging in to the ECS. For details, see **Logging In to a Windows ECS**.

Step 6 On the Windows remote desktop, use your browser to access Manager.

The address for accessing Manager is the address of the **MRS Manager** page. Enter the name and password of the cluster user, for example, user **admin**.

NOTE

- If you access Manager with other cluster usernames, change the password upon your first access. The new password must meet the requirements of the current password complexity policies. For details, contact the administrator.
- By default, a user is locked after inputting an incorrect password five consecutive times. The user is automatically unlocked after 5 minutes.

Step 7 Log out of FusionInsight Manager. To log out of Manager, move the cursor to  in the upper right corner and click **Log Out**.

----End

29.3 Using an MRS Client

29.3.1 Installing a Client

Scenario

This section describes how to install clients of all services (excluding Flume) in an MRS cluster. For details about how to install the Flume client, see **Installing the Flume Client**.

A client can be installed on a node inside or outside the cluster. This section uses the installation directory **//opt/client** as an example. Replace it with the actual one.

Prerequisites

- A Linux ECS has been prepared. For details about the supported OS of the ECS, see **Table 29-1**.

  Table 29-1 Reference list

CPU Architecture	OS	Supported Version
x86 computing	Euler	EulerOS 2.5
	SUSE	SUSE Linux Enterprise Server 12 SP4 (SUSE 12.4)
	Red Hat	Red Hat-7.5-x86_64 (Red Hat 7.5)
	CentOS	CentOS 7.6
Kunpeng computing (Arm)	Euler	EulerOS 2.8
	CentOS	CentOS 7.6

  In addition, sufficient disk space is allocated for the ECS, for example, 40 GB.
- The ECS and the MRS cluster are in the same VPC.
- The security group of the ECS must be the same as that of the master node in the MRS cluster.
- The NTP service has been installed on the ECS OS and is running properly.

If the NTP service is not installed, run the **yum install ntp -y** command to install it when the **yum** source is configured.

- A user can log in to the Linux ECS using the password (in SSH mode).
- All ports in the inbound direction of the MRS cluster security group are open to the client node. For details, see **Adding a Security Group Rule**.

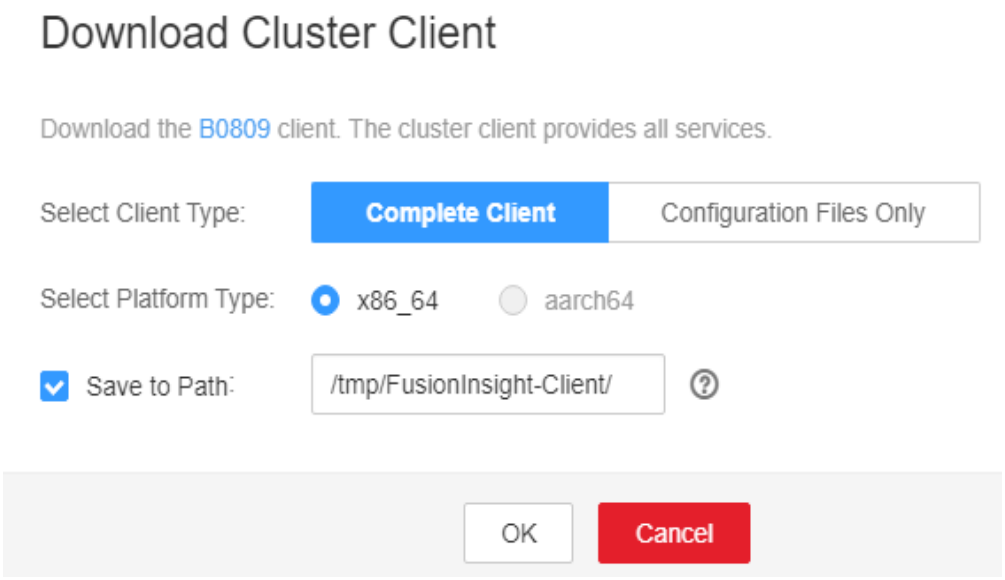
Installing a Client on a Node Inside a Cluster

1. Obtain the software package.

   Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**. Click the name of the cluster to be operated in the **Cluster** drop-down list.

   Choose **More > Download Client**. The **Download Cluster Client** dialog box is displayed.

   Figure 29-3 Downloading a client

   NOTE

   In the scenario where only one client is to be installed, choose **Cluster > Service > *Service name* > More > Download Client**. The **Download Client** dialog box is displayed.

2. Set the client type to **Complete Client**.

   Configuration Files Only is to download client configuration files in the following scenario: After a complete client is downloaded and installed and administrators modify server configurations on Manager, developers need to update the configuration files during application development.

   The platform type can be set to **x86_64** or **aarch64**.

   - **x86_64**: indicates the client software package that can be deployed on the x86 servers.
   - **aarch64**: indicates the client software package that can be deployed on the TaiShan servers.

> NOTE
>
> The cluster supports two types of clients: **x86_64** and **aarch64**. The client type must match the architecture of the node for installing the client. Otherwise, client installation will fail.

3. Select **Save to Path** and click **OK** to generate the client file.

   The generated file is stored in the **/tmp/FusionInsight-Client** directory on the active management node by default. You can also store the client file in a directory on which user **omm** has the read, write, and execute permissions. Copy the software package to the file directory on the server where the client is to be installed as user **omm** or **root**.

   The name of the client software package is in the follow format: **FusionInsight_Cluster_***<Cluster ID>***_Services_Client.tar**. In this section, the cluster ID **1** is used as an example. Replace it with the actual cluster ID.

   The following steps and sections use **FusionInsight_Cluster_1_Services_Client.tar** as an example.

   - Copy the client installation package to another directory on the current node, for example, **opt/Bigdata/client**.

     cp -p */tmp/FusionInsight-Client/* **FusionInsight_Cluster_1_Services_Client.tar** */opt/Bigdata/client*

   - Copy the client installation package to a directory on another node in the cluster, for example, **opt/Bigdata/client**.

     scp -p */tmp/FusionInsight-Client/* **FusionInsight_Cluster_1_Services_Client.tar** *IP address of the node where the client is to be installed:/opt/Bigdata/client*

   > NOTE
   >
   > If you cannot obtain the permissions of user **root**, use user **omm**.

4. Log in to the server where the client software package is located as user **user_client**.
5. Decompress the software package.

   Go to the directory where the installation package is stored, for example, **/opt/Bigdata/client**. Run the following command to decompress the installation package to a local directory:

   tar -xvf FusionInsight_Cluster_1_Services_Client.tar

6. Verify the software package.

   Run the following command to verify the decompressed file and check whether the command output is consistent with the information in the **sha256** file.

   sha256sum -c FusionInsight_Cluster_1_Services_ClientConfig.tar.sha256

   ```
   FusionInsight_Cluster_1_Services_ClientConfig.tar: OK
   ```

7. Decompress the obtained installation file.

   tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar

8. Go to the directory where the installation package is stored, and run the following command to install the client to a specified directory (an absolute path), for example, **/opt/client**:

   cd /opt/Bigdata/client/FusionInsight_Cluster_1_Services_ClientConfig

Run the **./install.sh /opt/client** command to install the client. The client is successfully installed if information similar to the following is displayed:

```
The component client is installed successfully
```

NOTE

- If the clients of all or some services use the **/opt/client** directory, other directories must be used when you install other service clients.
- You must delete the client installation directory when uninstalling a client.
- To ensure that an installed client can only be used by the installation user (for example, **user_client**), add parameter **-o** during the installation. That is, run the **./install.sh /opt/client -o** command to install the client.
- If an HBase client is installed, it is recommended that the client installation directory contain only uppercase and lowercase letters, digits, and characters (_-?.@+=) due to the limitation of the Ruby syntax used by HBase.

Using a Client

1. On the node where the client is installed, run the **sudo su - omm** command to switch the user. Run the following command to go to the client directory:

   cd /opt/client

2. Run the following command to configure environment variables:

   source bigdata_env

3. If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the user. If Kerberos authentication is disabled for the current cluster, skip this step.

   kinit *MRS cluster user*

   Example: **kinit admin**

   NOTE

   User **admin** is created by default for MRS clusters with Kerberos authentication enabled and is used for administrators to maintain the clusters.

4. Run the client command of a component directly.

   For example, run the **hdfs dfs -ls /** command to view files in the HDFS root directory.

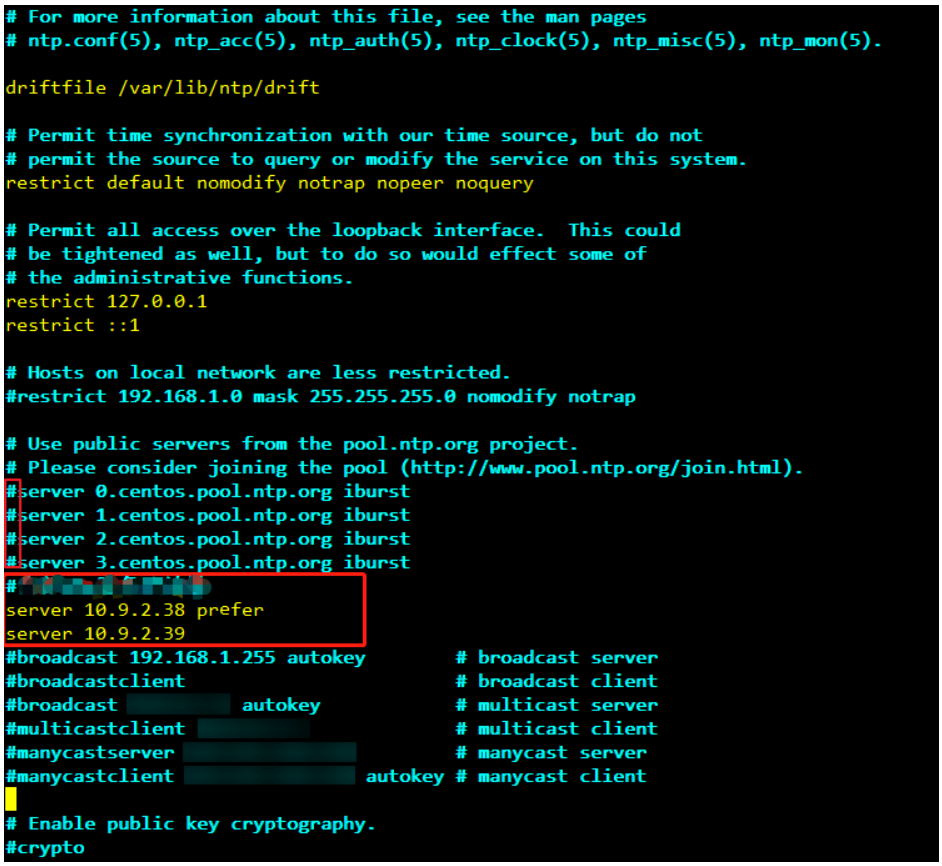
Installing a Client on a Node Outside a Cluster

1. Create an ECS that meets the requirements in **Prerequisites**.
2. Perform NTP time synchronization to synchronize the time of nodes outside the cluster with that of the MRS cluster.

   a. Run the **vi /etc/ntp.conf** command to edit the NTP client configuration file, add the IP addresses of the master node in the MRS cluster, and comment out the IP address of other servers.

   ```
   server master1_ip prefer
   server master2_ip
   ```


Figure 29-4 Adding the master node IP addresses



b. Run the **service ntpd stop** command to stop the NTP service.

c. Run the following command to manually synchronize the time:

/usr/sbin/ntpdate *192.168.10.8*

NOTE

192.168.10.8 indicates the IP address of the active Master node.

d. Run the **service ntpd start** or **systemctl restart ntpd** command to start the NTP service.

e. Run the **ntpstat** command to check the time synchronization result.

3. Perform the following steps to download the cluster client software package from FusionInsight Manager, copy the package to the ECS node, and install the client:

a. Log in to FusionInsight Manager and download the cluster client to the specified directory on the active management node by referring to **Accessing FusionInsight Manager** and **Installing a Client on a Node Inside a Cluster**.

b. Log in to the active management node as user **root** and run the following command to copy the client installation package to the target node:

scp -p /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Services_Client.tar *IP address of the node where the client is to be installed*:**/tmp**

c. Log in to the node on which the client is to be installed as the client user.

   Run the following commands to install the client. If the user does not have operation permissions on the client software package and client installation directory, grant the permissions using the **root** user.

   cd /tmp

   tar -xvf FusionInsight_Cluster_1_Services_Client.tar

   tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar

   cd FusionInsight_Cluster_1_Services_ClientConfig

   ./install.sh /opt/client

d. Run the following commands to switch to the client directory and configure environment variables:

   cd /opt/client

   source bigdata_env

e. If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the user. If Kerberos authentication is disabled for the current cluster, skip this step.

   kinit *MRS cluster user*

   Example: **kinit admin**

f. Run the client command of a component directly.

   For example, run the **hdfs dfs -ls /** command to view files in the HDFS root directory.

29.3.2 Updating a Client

A cluster provides a client for you to connect to a server, view task results, or manage data. If you modify service configuration parameters on Manager and restart the service, you need to download and install the client again or use the configuration file to update the client.

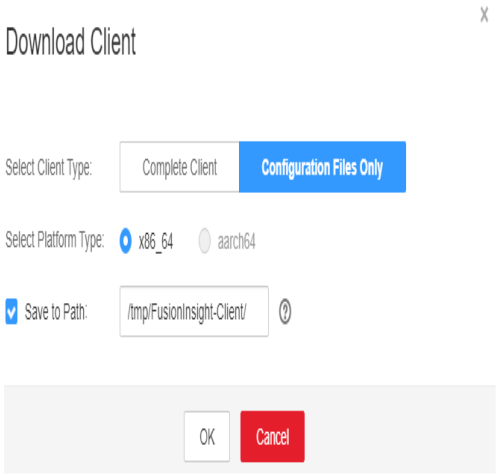
Updating the Client Configuration

Method 1:

Step 1 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**. Click the name of the cluster to be operated in the **Cluster** drop-down list.

Step 2 Choose **More** > **Download Client** > **Configuration Files Only**.

The generated compressed file contains the configuration files of all services.



Step 3 Determine whether to generate a configuration file on the cluster node.

- If yes, select **Save to Path**, and click **OK** to generate the client file. By default, the client file is generated in **/tmp/FusionInsight-Client** on the active management node. You can also store the client file in other directories, and user **omm** has the read, write, and execute permissions on the directories. Then go to **Step 4**.
- If no, click **OK**, specify a local save path, and download the complete client. Wait until the download is complete and go to **Step 4**.

Step 4 Use WinSCP to save the compressed file to the client installation directory, for example, **/opt/hadoopclient**, as the client installation user.

Step 5 Decompress the software package.

Run the following commands to go to the directory where the client is installed, and decompress the file to a local directory. For example, the downloaded client file is **FusionInsight_Cluster_1_Services_Client.tar**.

cd /opt/hadoopclient

tar -xvf FusionInsight_Cluster_1_Services_Client.tar

Step 6 Verify the software package.

Run the following command to verify the decompressed file and check whether the command output is consistent with the information in the **sha256** file.

sha256sum -c FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar.sha256

```
FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar: OK
```

Step 7 Decompress the package to obtain the configuration file.

tar -xvf FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar

Step 8 Run the following command in the client installation directory to update the client using the configuration file:

sh refreshConfig.sh *Client installation directory Directory where the configuration file is located*

For example, run the following command:

sh refreshConfig.sh /opt/hadoopclient /opt/hadoopclient/FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles

If the following information is displayed, the configurations have been updated successfully.

```
Succeed to refresh components client config.
```

----End

Method 2:

Step 1 Log in to the client installation node as user **root**.

Step 2 Go to the client installation directory, for example, **/opt/hadoopclient** and run the following commands to update the configuration file:

cd /opt/hadoopclient

sh autoRefreshConfig.sh

Step 3 Enter the username and password of the FusionInsight Manager administrator and the floating IP address of FusionInsight Manager.

Step 4 Enter the names of the components whose configuration needs to be updated. Use commas (,) to separate the component names. Press **Enter** to update the configurations of all components if necessary.

If the following information is displayed, the configurations have been updated successfully.

```
Succeed to refresh components client config.
```

----End

