MapReduce Service

Component Operation Guide

Issue 01

Date 2024-12-13

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions

Notice

Contents

1 Using Alluxio

1.1 Configuring an Underlying Storage System

If you want to use a unified client API and a global namespace to access persistent storage systems including HDFS and OBS to separate computing from storage, you can configure the underlying storage system of Alluxio on MRS Manager. After a cluster is created, the default underlying storage address is **hdfs://hacluster/**, that is, the HDFS root directory is mapped to Alluxio.

Prerequisites

- Alluxio has been installed in a cluster.
- The password of user **admin** has been obtained. The password of user **admin** is specified by the user during MRS cluster creation.

Configuring HDFS as the Underlying File System of Alluxio

NOTE

Security clusters with Kerberos authentication enabled do not support this function.

Step 1 Go to the **All Configurations** page of Alluxio. See **Modifying Cluster Service Configuration Parameters**.

Step 2 In the left pane, choose **Alluxio** > **Under Stores**, and modify the value of **alluxio.master.mount.table.root.ufs** to **hdfs://hacluster***/XXX/*.

For example, if you want to use *HDFS root directory***/alluxio/** as the root directory of Alluxio, modify the value of **alluxio.master.mount.table.root.ufs** to **hdfs://hacluster/alluxio/**.

Step 3 Click **Save Configuration**. In the displayed dialog box, select **Restart the affected services or instances**.

Step 4 Click **OK** to restart Alluxio.

----End

Configuring Huawei OBS as the Underlying File System of Alluxio

Method 1:

Step 1 Configure an agency with the **OBS OperateAccess** permission for the cluster. For details, see **Configuring a Storage-Compute Decoupled Cluster (Agency)**.

Step 2 Go to the **All Configurations** page of Alluxio. See **Modifying Cluster Service Configuration Parameters**.

Step 3 In the left pane, choose **Alluxio** > **Under Stores**, and modify the value of **alluxio.master.mount.table.root.ufs** to **obs://<OBS_BUCKET>/<OBS_DIRECTORY>/**. **OBS_BUCKET** is an existing OBS file system name, and **OBS_DIRECTORY** is a directory in the file system.

Step 4 Click **Save Configuration**. In the displayed dialog box, select **Restart the affected services or instances**.

Step 5 Click **OK** to restart Alluxio.

----End

Method 2:

Step 1 Configure an agency with the **OBS OperateAccess** permission for the cluster. For details, see **Configuring a Storage-Compute Decoupled Cluster (Agency)**.

Step 2 Log in to the active Master node. For details about how to determine the active and standby management nodes, see **Determining Active or Standby Management Nodes of MRS Manager**.

Step 3 Run the following command to configure environment variables:

source /opt/client/bigdata_env

NOTE

In the preceding command, **/opt/client** is the directory where current cluster client is installed. Change it based on the site requirements.

Step 4 Run the following command to mount the internal directory of an OBS container to the **/obs** directory of Alluxio:

alluxio fs mount /obs obs://<OBS_BUCKET>/<OBS_DIRECTORY>/

----End

1.2 Accessing Alluxio Using a Data Application

The port number used for accessing the Alluxio file system is 19998, and the access address is **alluxio://***<Master node IP address of Alluxio>***:19998/***<PATH>*. This section uses examples to describe how to access the Alluxio file system using data applications (Spark, Hive, Hadoop MapReduce, and Presto).

Using Alluxio as the Input and Output of a Spark Application

Step 1 Log in to the Master node in a cluster as user **root** using the password set during cluster creation.

Step 2 Run the following command to configure environment variables:

source /opt/client/bigdata_env

Step 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**

Step 4 Prepare an input file and copy local data to the Alluxio file system.

For example, prepare the input file **test_input.txt** in the local **/home** directory, and run the following command to save the **test_input.txt** file to Alluxio:

alluxio fs copyFromLocal /home/test_input.txt /input

Step 5 Run the following commands to start **spark-shell**:

spark-shell

Step 6 Run the following commands in spark-shell:

val s = sc.textFile("alluxio://<*Name of the Alluxio node*>:19998/input")

val double = s.map(line => line + line)

double.saveAsTextFile("alluxio://<*Name of the Alluxio node*>:19998/output")

NOTE

Replace ***Name of the Alluxio node*>:19998** with the actual node name and port numbers of all nodes where the AlluxioMaster instance is deployed. Use commas (,) to separate the node name and port number, for example, **node-ana-coremspb.mrs-m0va.com:19998,node-master2kiww.mrs-m0va.com:19998,node-master1cqwv.mrs-m0va.com:19998**.

Step 7 Press **Ctrl+C** to exit spark-shell.

Step 8 Run the **alluxio fs ls /** command to check whether the output directory **/output** containing double content of the input file exists in the root directory of Alluxio.

----End

Creating a Hive Table on Alluxio

Step 1 Log in to the Master node in a cluster as user **root** using the password set during cluster creation.

Step 2 Run the following command to configure environment variables:

source /opt/client/bigdata_env

Step 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**

Step 4 Prepare an input file. For example, prepare the **hive_load.txt** input file in the local **/home** directory. The file content is as follows:

```
1, Alice, company A
2, Bob, company B
```

Step 5 Run the following command to import the **hive_load.txt** file to Alluxio:

alluxio fs copyFromLocal /home/hive_load.txt /hive_input

Step 6 Run the following command to start the Hive beeline:

beeline

Step 7 Run the following commands in beeline to create a table based on the input file in Alluxio:

CREATE TABLE u_user(id INT, name STRING, company STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

LOAD DATA INPATH 'alluxio://<*Name of the Alluxio node*>:19998/hive_input' INTO TABLE u_user;

NOTE

Replace ***Name of the Alluxio node*>:19998** with the actual node name and port numbers of all nodes where the AlluxioMaster instance is deployed. Use commas (,) to separate the node name and port number, for example, **node-ana-coremspb.mrs-m0va.com:19998,node-master2kiww.mrs-m0va.com:19998,node-master1cqwv.mrs-m0va.com:19998**.

Step 8 Run the following command to view the created table:

select * from u_user;

----End

Running Hadoop Wordcount in Alluxio

Step 1 Log in to the Master node in a cluster as user **root** using the password set during cluster creation.

Step 2 Run the following command to configure environment variables:

source /opt/client/bigdata_env

Step 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**

Step 4 Prepare an input file and copy local data to the Alluxio file system.

For example, prepare the input file **test_input.txt** in the local **/home** directory, and run the following command to save the **test_input.txt** file to Alluxio:

alluxio fs copyFromLocal /home/test_input.txt /input

Step 5 Run the following command to execute the wordcount job:

yarn jar /opt/share/hadoop-mapreduce-examples-<*Hadoop version*>-mrs-<*MRS cluster version*>/hadoop-mapreduce-examples-<*Hadoop version*>-mrs-<*MRS cluster version*>.jar wordcount alluxio://<*Name of the Alluxio node*>:19998/input alluxio://<*Name of the Alluxio node*>:19998/output

NOTE

- Replace **<*Hadoop version*>** with the actual one.
- Replace **<*MRS cluster version*>** with the major version of MRS. For example, for a cluster of MRS 1.9.2, mrs-1.9.0 is used.
- Replace ***Name of the Alluxio node*>:19998** with the actual node name and port numbers of all nodes where the AlluxioMaster instance is deployed. Use commas (,) to separate the node name and port number, for example, **node-ana-coremspb.mrs-m0va.com:19998,node-master2kiww.mrs-m0va.com:19998,node-master1cqwv.mrs-m0va.com:19998**.

Step 6 Run the **alluxio fs ls /** command to check whether the output directory **/output** containing the wordcount result exists in the root directory of Alluxio.

----End

Using Presto to Query Tables in Alluxio

Step 1 Log in to the Master node in a cluster as user **root** using the password set during cluster creation.

Step 2 Run the following command to configure environment variables:

source /opt/client/bigdata_env

Step 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**

Step 4 Run the following commands to start Hive Beeline to create a table on Alluxio.

beeline

CREATE TABLE u_user (id int, name string, company string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 'alluxio://<*Name of the Alluxio node*>:19998/u_user';

insert into u_user values(1,'Alice','Company A'),(2, 'Bob', 'Company B');

NOTE

Replace ***Name of the Alluxio node*>:19998** with the actual node name and port numbers of all nodes where the AlluxioMaster instance is deployed. Use commas (,) to separate the node name and port number, for example, **node-ana-coremspb.mrs-m0va.com:19998,node-master2kiww.mrs-m0va.com:19998,node-master1cqwv.mrs-m0va.com:19998**.

Step 5 Start the Presto client. For details, see **Step 2** to **Step 8** in **Using a Client to Execute Query Statements**.

Step 6 On the Presto client, run the **select * from hive.default.u_user;** statement to query the table created in Alluxio:

Figure 1-1 Using Presto to query the table created in Alluxio

```
presto> select * from hive.default.u_user;
 id | name  |  company
----+-------+-----------
  1 | Alice | Company A
  2 | Bob   | Company B
(2 rows)
```

----End

1.3 Common Operations of Alluxio

Preparations

1. Create a cluster with Alluxio installed.
2. Log in to the active Master node in a cluster as user **root** using the password set during cluster creation.
3. Run the following command to configure environment variables:

   source /opt/client/bigdata_env

Using the Alluxio Shell

The **Alluxio shell** contains multiple command line operations that interact with Alluxio.

- View a file system operation command list:

  alluxio fs
- Run the **ls** command to list the files in Alluxio. For example, list all files in the root directory:

  alluxio fs ls /
- Run the **copyFromLocal** command to copy local files to Alluxio:

  alluxio fs copyFromLocal /home/test_input.txt /test_input.txt

  Command output:
  ```
  Copied file:///home/test_input.txt to /test_input.txt
  ```
- Run the **ls** command again to list the files in Alluxio. The copied **test_input.txt** file is listed:

  alluxio fs ls /

  Command output:
  ```
  12      PERSISTED 11-28-2019 17:10:17:449 100% /test_input.txt
  ```
  The **test_input.txt** file is displayed in Alluxio. The parameters in the file indicate the file size, whether the file is persistent, creation date, cache ratio of the file in Alluxio, and file name.
- Run the **cat** command to print file content:

  alluxio fs cat /test_input.txt

  Command output:

```
Test Alluxio
```

Mounting Function of Alluxio

Alluxio uses a unified namespace feature to unify the access to storage systems. For details, see **https://docs.alluxio.io/os/user/2.0/en/advanced/Namespace-Management.html**.

This feature allows users to mount different storage systems to an Alluxio namespace and seamlessly access files across storage systems through the Alluxio namespace.

1. Create a directory as a mount point in Alluxio.

   alluxio fs mkdir /mnt
   ```
   Successfully created directory /mnt
   ```
2. Mount an existing OBS file system to Alluxio. (Prerequisite: An agency with the **OBS OperateAccess** permission has been configured for the cluster. For details, see **Configuring a Storage-Compute Decoupled Cluster (Agency)**.) The **obs-mrstest** file system is used as an example. Replace the file system name with the actual one.

   alluxio fs mount /mnt/obs obs://obs-mrstest/data
   ```
   Mounted obs://obs-mrstest/data at /mnt/obs
   ```
3. List files in the OBS file system using the Alluxio namespace. Run the **ls** command to list the files in the OBS mount directory.

   alluxio fs ls /mnt/obs
   ```
   38      PERSISTED 11-28-2019 17:42:54:554   0% /mnt/obs/hive_load.txt
   12      PERSISTED 11-28-2019 17:43:07:743   0% /mnt/obs/test_input.txt
   ```
   You can also view the newly mounted files and directories on the Alluxio web UI.
4. Use the unified namespace of Alluxio to seamlessly exchange data between different storage systems. For example, run the **ls -R** command to list all files in a directory recursively:

   alluxio fs ls -R /
   ```
          0       PERSISTED 11-28-2019 11:15:19:719  DIR /app-logs
          1       PERSISTED 11-28-2019 11:18:36:885  DIR /apps
          1       PERSISTED 11-28-2019 11:18:40:209  DIR /apps/templeton
   239440292       PERSISTED 11-28-2019 11:18:40:209   0% /apps/templeton/hive.tar.gz
   .....
          1       PERSISTED 11-28-2019 19:00:23:879  DIR /mnt
          2       PERSISTED 11-28-2019 19:00:23:879  DIR /mnt/obs
         38       PERSISTED 11-28-2019 17:42:54:554   0% /mnt/obs/hive_load.txt
         12       PERSISTED 11-28-2019 17:43:07:743   0% /mnt/obs/test_input.txt
   .....
   ```
   The command output shows all files that are from the mounted storage system in the root directory of the Alluxio file system (the default directory is the HDFS root directory, that is, **hdfs://hacluster/**). The **/app-logs** and **/apps** directories are in HDFS, and the **/mnt/obs/** directory is in OBS.

Using Alluxio to Accelerate Data Access

Alluxio can accelerate data access, because it uses memory to store data. Example commands are provided as follows:

1. Upload the **test_data.csv** file (a sample that records recipes) to the **/data** directory of the **obs-mrstest** file system. Run the **ls** command to display the file status.

   alluxio fs ls /mnt/obs/test_data.csv
   ```
   294520189        PERSISTED 11-28-2019 19:38:55:000   0% /mnt/obs/test_data.csv
   ```
   The output indicates that the cache percentage of the file in Alluxio is 0%, that is, the file is not in Alluxio memory.

2. Count the occurrence times of the word "milk" in the file, and calculate the time consumed.

   time alluxio fs cat /mnt/obs/test_data.csv | grep -c milk
   ```
   52180

   real    0m10.765s
   user    0m5.540s
   sys     0m0.696s
   ```

3. Data is stored in memory after being read for the first time. When Alluxio reads data again, the data access speed is increased. For example, after running the **cat** command to obtain a file, run the **ls** command to check the file status.

   alluxio fs ls /mnt/obs/test_data.csv
   ```
   294520189        PERSISTED 11-28-2019 19:38:55:000 100% /mnt/obs/test_data.csv
   ```
   The output shows that the file has been fully loaded to Alluxio.

4. Access the file again, count the occurrence times of the word "eggs", and calculate the time consumed.

   time alluxio fs cat /mnt/obs/test_data.csv | grep -c eggs
   ```
   59510

   real    0m5.777s
   user    0m5.992s
   sys     0m0.592s
   ```
   According to the comparison of the two time consumption records, the time consumed for accessing data stored in Alluxio memory is significantly reduced.

2 Using CarbonData (for Versions Earlier Than MRS 3.*x*)

2.1 Using CarbonData from Scratch

This section is for MRS 3.x or earlier. For MRS 3.x or later, see **Using CarbonData (for MRS 3.x or Later)**.

This section describes the procedure of using Spark CarbonData. All tasks are based on the Spark-beeline environment. The tasks include:

1. Connecting to Spark

   Before performing any operation on CarbonData, users must connect CarbonData to Spark.

2. Creating a CarbonData table

   After connecting to Spark, users must create a CarbonData table to load and query data.

3. Loading data to the CarbonData table

   Users load data from CSV files in HDFS to the CarbonData table.

4. Querying data from the CarbonData table

   After data is loaded to the CarbonData table, users can run query commands such as **groupby** and **where**.

Prerequisites

A client has been installed. For details, see **Using an MRS Client**.

Procedure

Step 1 Connect CarbonData to Spark.

1. Prepare a client based on service requirements and use user **root** to log in to the node where the client is installed.

   For example, if you have updated the client on the Master2 node, log in to the Master2 node to use the client. For details, see **Using an MRS Client**.

2. Run the following commands to switch the user and configure environment variables:

   sudo su - omm

   source /opt/client/bigdata_env

3. For clusters with Kerberos authentication enabled, run the following command to authenticate the user. For clusters with Kerberos authentication disabled, skip this step.

   kinit *Spark username*

   NOTE

   The user needs to be added to user groups **hadoop** (primary group) and **hive**.

4. Run the following command to connect to the Spark environment.

   spark-beeline

Step 2 Create a CarbonData table.

Run the following command to create a CarbonData table, which is used to load and query data.

CREATE TABLE x1 (imei string, deviceInformationId int, mac string, productdate timestamp, updatetime timestamp, gamePointId double, contractNumber double)

STORED BY 'org.apache.carbondata.format'

TBLPROPERTIES ('DICTIONARY_EXCLUDE'='mac','DICTIONARY_INCLUDE'='deviceInformationId');

The command output is as follows:

```
+---------+--+
| result  |
+---------+--+
+---------+--+
No rows selected (1.551 seconds)
```

Step 3 Load data from CSV files to the CarbonData table.

Run the command to load data from CSV files based on the required parameters. Only CSV files are supported. The CSV column name and sequence configured in the **LOAD** command must be consistent with those in the CarbonData table. The data formats and number of data columns in the CSV files must also be the same as those in the CarbonData table.

The CSV files must be stored on HDFS. You can upload the files to OBS and import them from OBS to HDFS on the **Files** page of the MRS console. For details, see **Importing and Exporting Data**.

If Kerberos authentication is enabled, prepare the CSV files in the work environment and import them to HDFS using open-source HDFS commands. In addition, assign the Spark user with the read and execute permissions of the files on HDFS by referring to **5**.

For example, the **data.csv** file is saved in the **tmp** directory of HDFS with the following contents:

```
x123,111,dd,2017-04-20 08:51:27,2017-04-20 07:56:51,2222,33333
```


The command for loading data from that file is as follows:

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

The command output is as follows:

```
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (3.039 seconds)
```

Step 4 Query data from the CarbonData.

- **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 the where condition**

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

  select * from x1 where deviceinformationid='111';

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.

Step 5 Run the following command to exit the Spark environment.

!quit

----End

2.2 About CarbonData Table

Description

CarbonData tables are similar to tables in the relational database management system (RDBMS). RDBMS tables consist of rows and columns to store data. CarbonData tables have fixed columns and also store structured data. In CarbonData, data is saved in entity files.

Supported Data Types

CarbonData tables support the following data types:

- Int
- String
- BigInt
- Decimal
- Double
- TimeStamp

Table 2-1 describes the details about each data type.

Table 2-1 CarbonData data types

Data Type	Description
Int	4-byte signed integer ranging from -2,147,483,648 to 2,147,483,647 **NOTE** If a non-dictionary column is of the **int** data type, it is internally stored as the **BigInt** type.
String	The maximum character string length is 100000.
BigInt	Data is saved using the 64-bit technology. The value ranges from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
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	Data is saved using the 64-bit technology. The value ranges from 4.9E-324 to 1.7976931348623157E308.
TimeStamp	**yyyy-MM-dd HH:mm:ss** format is used by default.

NOTE

Measurement of all Integer data is processed and displayed using the **BigInt** data type.

2.3 Creating a CarbonData Table

Scenario

A CarbonData table must be created to load and query data.

Creating a Table with Self-Defined Columns

Users can create a table by specifying its columns and data types. For analysis clusters with Kerberos authentication enabled, if a user wants to create a CarbonData table in a database other than the **default** database, the **Create** permission of the database must be added to the role to which the user is bound in Hive role management.

Sample command:

CREATE TABLE IF NOT EXISTS productdb.productSalesTable (

productNumber Int,

productName String,

storeCity String,

storeProvince String,

revenue Int)

STORED BY 'org.apache.carbondata.format'

TBLPROPERTIES (

'table_blocksize'='128',

'DICTIONARY_EXCLUDE'='productName',

'DICTIONARY_INCLUDE'='productNumber');

The following table describes parameters of preceding commands.

Table 2-2 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 (_).
productNumber productName storeCity storeProvince revenue	Columns in the table. The columns are service entities for data analysis. The column name (field name) consists of letters, digits, and underscores (_). **NOTE** In CarbonData, you cannot configure a column's NOT NULL or default value, or the primary key of the table.

Parameter	Description
table_blocksize	Block size of data files used by the CarbonData table. The value ranges from 1 MB to 2048 MB. The default is 1024 MB. ● If the value of **table_blocksize** is too small, a large number of small files will be generated when data is loaded. This may affect the performance in using HDFS. ● If the value of **table_blocksize** is too large, a large volume of data must be read from a block and the read concurrency is low when data is queried. As a result, the query performance deteriorates. 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.
DICTIONARY_EXCLUDE	Specifies the columns that do not generate dictionaries. This function is optional and applicable to columns of high complexity. By default, the system generates dictionaries for columns of the String type. However, as the number of values in the dictionaries increases, conversion operations by the dictionaries increase and the system performance deteriorates. Generally, if a column has over 50,000 unique data records, it is considered as a highly complex column and dictionary generation must be disabled. **NOTE** Non-dictionary columns support only the String and Timestamp data types.
DICTIONARY_INCLUDE	Specifies the columns that generate dictionaries. This function is optional and applicable to columns of low complexity. It improves the performance of queries with the **groupby** condition. Generally, the complexity of a dictionary column cannot exceed 50,000.

2.4 Deleting a CarbonData Table

Scenario

Unused CarbonData tables can be deleted. After a CarbonData table is deleted, its metadata and loaded data are deleted together.

Procedure

Step 1 Run the following command to delete a CarbonData table:

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

db_name is optional. If **db_name** is not specified, the table named **table_name** in the current database is deleted.

For example, run the following command to delete the **productSalesTable** table in the **productdb** database:

DROP TABLE productdb.productSalesTable;

Step 2 Run the following command to confirm that the table is deleted:

SHOW TABLES;

----End

3 Using CarbonData (for MRS 3.*x* or Later)

3.1 CarbonData Data Types

Overview

In CarbonData, data is stored in entities called tables. CarbonData tables are similar to RDBMS tables. RDBMS data is stored in a table consisting of 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 range.

Table 3-1 CarbonData data types

Data Type	Value Range
Int	4-byte signed integer ranging from -2,147,483,648 to 2,147,483,647. **NOTE** If a non-dictionary column is of the **int** data type, it 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 3-2 Main specifications of CarbonData

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 Strings: 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.

The memory required for data loading depends on the following factors:

- Number of columns
- Column values
- Concurrency (configured using **carbon.number.of.cores.while.loading**)
- Sort size in memory (configured using **carbon.sort.size**)
- Intermediate cache (configured using **carbon.graph.rowset.size**)

Data loading of an 8 GB CSV file that contains 10 million records and 300 columns with each row size being about 0.8 KB requires about 10 GB executor memory.

That is, set **carbon.sort.size** to **100000** and retain the default values for other parameters.

Specifications of Secondary Index Tables

Table 3-3 Table specifications

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

3.2 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 3-21** or **Table 3-22** have been configured.

Hive ACL permissions

Table 3-4 Hive ACL permissions required for CarbonData table-level operations

Scenario	Required Permission
DESCRIBE TABLE	SELECT (of table)
SELECT	SELECT (of table)

Scenario	Required Permission
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)
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.

3.3 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 3-5 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

   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 the CarbonData table, and query data in CarbonData 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()
```

3.4 CarbonData Data Analytics

3.4.1 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 3-6 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.

3.4.2 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**.

3.4.3 Modify the CarbonData Table

SET and UNSET

When the **SET** command is executed, the new properties overwrite the existing ones.

- SORT SCOPE

  The following is an example of the **SET SORT SCOPE** command:

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

  After running the **UNSET SORT SCOPE** command, the default value **NO_SORT** is adopted.

  The following is an example of the **UNSET SORT SCOPE** command:

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

- SORT COLUMNS

  The following is an example of the **SET SORT COLUMNS** command:

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

After this command is executed, the new value of **SORT_COLUMNS** is used. Users can adjust the **SORT_COLUMNS** based on the query results, but the original data is not affected. The operation does not affect the query performance of the original data segments which are not sorted by new **SORT_COLUMNS**.

The **UNSET** command is not supported, but the **SORT_COLUMNS** can be set to empty string instead of using the **UNSET** command.

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

NOTE

- The later version will enhance custom compaction to resort the old segments.
- The value of **SORT_COLUMNS** cannot be modified in the streaming table.
- If the **inverted index** column is removed from **SORT_COLUMNS**, **inverted index** will not be created in this column. However, the old configuration of **INVERTED_INDEX** will be kept.

3.4.4 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 about CarbonData parameters, see **Typical CarbonData Configuration Parameters**.

3.4.5 Deleting CarbonData Table Segments

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.

3.4.6 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 compacted and sorted, and multiple segments are compacted 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 3-7 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.
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.

Parameter	Default Value	Application Type	Description
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.
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.

Parameter	Default Value	Application Type	Description
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?**.

3.5 CarbonData Performance Tuning

3.5.1 CarbonData 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 3-18**.

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 3-8 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.

3.5.2 Typical CarbonData Performance Tuning Parameters

Scenario

This section describes the configurations that can improve CarbonData performance.

Procedure

Table 3-9 and **Table 3-10** describe the configurations about query of CarbonData.

Table 3-9 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 3-10 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 3-11, **Table 3-12**, and **Table 3-13** describe the configurations for CarbonData data loading.

Table 3-11 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 3-12 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 3-13 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 3-14 describes the configurations for CarbonData data loading and query.

Table 3-14 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 3-15 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**.

3.5.3 Suggestions for Creating CarbonData Tables

Scenario

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

Table 3-16 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
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');
```

3.6 Typical CarbonData Configuration Parameters

This section provides the details of all the configurations required for the CarbonData System.

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** > **Spark2x**. 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 3-17 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**.

Parameter	Default Value	Description
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.
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.

Table 3-18 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.

Parameter	Default Value	Description
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
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.
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.

Parameter	Default Value	Description
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.
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.

Parameter	Default Value	Description
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.
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		
carbon.number.of.cores	4	Number of cores to be used during query

Parameter	Default Value	Description
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 CarbonData 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 sub-sentence query returns 10,000 or fewer records, enabling this parameter will give the 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.
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		

Parameter	Default Value	Description
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 3-19 Other configurations in **carbon.properties**

Parameter	Default Value	Description
Data Loading Configuration		
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 CarbonData Performance Tuning Parameters**.

Parameter	Default Value	Description
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.
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 CarbonData 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		

Parameter	Default Value	Description
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.
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.
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 3-20** in the *{Client installation directory}*/**Spark/spark/conf/spark-defaults.conf** file.

Table 3-20 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 3-21** in the *{Client installation directory}*/**Spark/spark/conf/spark-defaults.conf** file.

Table 3-21 Parameter description

Parameter	Value	Description
spark.driver.extraJavaOptions	-Dlog4j.configuration=file:/opt/client/Spark2x/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/Spark2x/spark/conf/jaas.conf -Dorg.xerial.snappy.tempdir=/opt/client/Spark2x/tmp -Dcarbon.properties.filepath=/opt/client/Spark2x/spark/conf/carbon.properties -Djava.io.tmpdir=/opt/client/Spark2x/tmp	The default value **/opt/client/Spark2x/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.CarbonInternalSqlAstBuilder	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.

– In JDBCServer: Log in to the node where JDBCServer is installed and configure the parameters listed in Table 3-22 in the **{BIGDATA_HOME}/FusionInsight_Spark_*/*_JDBCServer/etc/spark-defaults.conf** file.

Table 3-22 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/spark2x/JDBCServer/snappy_tmp -Djava.io.tmpdir=${BIGDATA_HOME}/tmp/spark2x/JDBCServer/io_tmp -Dcarbon.properties.filepath=${SPARK_CONF_DIR}/carbon.properties -	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
	Djdk.tls.ephemeralDHKeySize=2048 -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.

3.7 CarbonData Syntax Reference

3.7.1 DDL

3.7.1.1 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 3-23 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).

    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.
  - **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.

3.7.1.2 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 3-24 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.

3.7.1.3 DROP TABLE

Function

This command is used to delete an existing table.

Syntax

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

Parameter Description

Table 3-25 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.

3.7.1.4 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 3-26 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.

3.7.1.5 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 3-27 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 3-7**.
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**.

3.7.1.6 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 3-28 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.

3.7.1.7 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 3-29 ADD COLUMNS parameters

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

Parameter	Description
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.

3.7.1.8 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 3-30 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.

3.7.1.9 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*;

Parameters

Table 3-31 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.

  Example:

  - 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 Value

- 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.

3.7.1.10 REFRESH TABLE

Function

This command is used to register Carbon table to Hive meta store catalogue from exisiting Carbon table data.

Syntax

REFRESH TABLE *db_name.table_name*;

Parameter Description

Table 3-32 REFRESH TABLE parameters

Parameter	Description
db_name	Name of the database. If this parameter is left unspecified, the current database is selected.
table_name	Name of the table.

Usage Guidelines

- The new database name and the old database name should be same.
- Before executing this command the old table schema and data should be copied into the new database location.
- If the table is aggregate table, then all the aggregate tables should be copied to the new database location.
- For old store, the time zone of the source and destination cluster should be same.
- If old cluster used HIVE meta store to store schema, refresh will not work as schema file does not exist in file system.

Examples

REFRESH TABLE *dbcarbon.productSalesTable*;

System Response

By running this command, the Carbon table will be registered to Hive meta store catalogue from exisiting Carbon table data.

3.7.1.11 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 3-33 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.

3.7.2 DML

3.7.2.1 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 3-34 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 3-35**.

Table 3-35 Bad Records Logger

Configuration Item	Default Value	Description
BAD_RECORDS_LOGGER_ENABLE	false	Whether to create logs with 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 3-36 MAXCOLUMNS

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

Table 3-37 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.

    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.

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.

3.7.2.2 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 3-38 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.

3.7.2.3 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 3-39 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.

3.7.2.4 INSERT INTO CARBON TABLE

Function

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

Syntax

INSERT INTO [CARBON TABLE] [select query];

Parameter Description

Table 3-40 INSERT INTO parameters

Parameter	Description
CARBON TABLE	Name of the CarbonData table 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.

3.7.2.5 DELETE SEGMENT by ID

Function

This command is used to delete segments by the ID.

Syntax

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

Parameter Description

Table 3-41 DELETE SEGMENT parameters

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

Usage Guidelines

Segments cannot be deleted from the stream table.

Examples

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

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

System Response

Success or failure will be recorded in the CarbonData log.

3.7.2.6 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 3-42 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.

3.7.2.7 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 3-43 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  |
+-----+----------+-------------------------+-----------------+------------+------------+-------------+--------------+--
+
```

3.7.2.8 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 3-44 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 (_).

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

3.7.2.9 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 3-45 SHOW SECONDARY INDEXES parameters

Parameter	Description
db_name	Database name. It consists of letters, digits, and special characters (_).

Parameter	Description
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.

3.7.2.10 DROP SECONDARY INDEX

Function

This command is used to delete the existing secondary index table in a specific table.

Syntax

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

Parameter Description

Table 3-46 DROP SECONDARY INDEX parameters

Parameter	Description
index_name	Name of the index table. Table name contains letters, digits, and underscores (_).
db_Name	Name of the database. If the parameter is not specified, the current database is used.
table_name	Name of the table to be deleted.

Usage Guidelines

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

Examples

DROP INDEX *if exists productNameIndexTable* **ON** *productdb.productSalesTable*;

System Response

Secondary Index Table will be deleted. Index information will be cleared in CarbonData table and the success message will be recorded in system logs.

3.7.2.11 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 3-47 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.

3.7.2.12 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 3-48 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 3-49 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.

Property	Description
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.
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

- Success will be recorded in the driver log.
- Failure will be displayed on the UI.

3.7.3 Concurrent CarbonData Table Operations

Before performing **DDL** and **DML** operations, you need to obtain the corresponding locks. See **Table 3-50** 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 3-50** 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 3-50 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	√	-	√	-	-	-	-	√	-	-

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
ALTER TABLE COMPACTION	-	√	-	-	-	-	√	-	-	-
TABLE RENAME	-	-	-	-	-	-	-	-	-	-
ADD COLUMNS	√	√	-	-	-	-	-	-	-	-
DROP COLUMNS	√	√	-	-	-	-	-	-	-	-
CHANGE DATA TYPE	√	√	-	-	-	-	-	-	-	-
REFRESH TABLE	-	-	-	-	-	-	-	-	-	-
REGISTER INDEX TABLE	√	-	-	-	-	-	-	-	-	-
REFRESH 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
LOAD DATA/ INSERT INTO	-	-	-	-	-	-	-	-	√	√
UPDATE CARBON TABLE	√	√	-	-	-	-	√	-	-	-
DELETE RECORDS from CARBON TABLE	√	√	-	-	-	-	√	-	-	-
DELETE SEGMENT by ID	-	-	-	√	√	-	-	-	-	-
DELETE SEGMENT by DATE	-	-	-	√	√	-	-	-	-	-
SHOW SEGMENTS	-	-	-	-	-	-	-	-	-	-

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 SECONDARY INDEX	√	√	-	√	-	-	-	-	-	-
SHOW SECONDARY INDEXES	-	-	-	-	-	-	-	-	-	-
DROP SECONDARY INDEX	√	-	√	-	-	-	-	-	-	-
CLEAN FILES	-	-	-	-	-	-	-	-	-	-
SET/RESET	-	-	-	-	-	-	-	-	-	-
Add Hive Partition	-	-	-	-	-	-	-	-	-	-
Drop Hive Partition	√	√	√	√	√	√	-	-	-	-
Drop Partition	√	√	√	√	√	√	-	-	-	-

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
Alter table set	√	√	-	-	-	-	-	-	-	-

3.7.4 CarbonData Segment APIs

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)
```

3.7.5 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

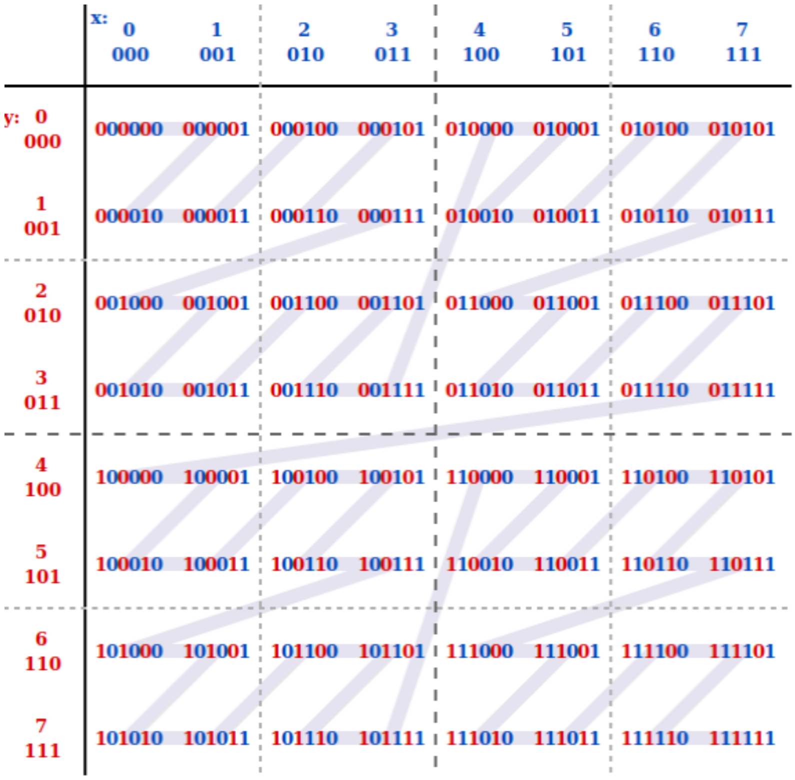
Spatial data includes multidimensional points, lines, rectangles, cubes, polygons, and other geometric objects. A spatial data object occupies a certain region of space, called spatial scope, characterized by its location and boundary. The spatial data can be either point data or region data.

- 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 3-51 Parameter description

Parameter	Description
SPATIAL_INDEX	Specifies the spatial index. Its value is the same as the column name.

Parameter	Description
SPATIAL_INDEX.xxx.type	(Mandatory) The value is set to **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))`

Parameter	Type	Description
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.
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;
```

3.8 CarbonData Troubleshooting

3.8.1 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.

3.8.2 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

3.9 CarbonData FAQs

3.9.1 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 |
+------+---------------------+--+
```


3.9.2 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.

3.9.3 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**.

3.9.4 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;*

3.9.5 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 ****.

3.9.6 Why Data Load Performance Decreases due to Bad Records?

Question

Why data load performance decreases due to bad records?

Answer

If bad records are present in the data and **BAD_RECORDS_LOGGER_ENABLE** is **true** or **BAD_RECORDS_ACTION** is **redirect** then load performance will decrease due to extra I/O for writing failure reason in log file or redirecting the records to raw CSV.

3.9.7 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.

3.9.8 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.

3.9.9 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. So you need to increase the memory by increasing the value of the parameter **spark.yarn.am.memoryOverhead**.

3.9.10 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;

3.9.11 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*;

3.9.12 Why Does the Missing Privileges Exception Occur When the Database Is Dropped?

Question

Why does the following exception occur when the database is dropped?

```
Error: org.apache.spark.sql.AnalysisException: Missing Privileges;(State=,code=0)
```

Answer

This error occurs when the database owner runs the ***drop database <database_name> cascade*** command and tables created by other users are included.

3.9.13 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

3.9.14 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 off 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**.

3.9.15 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.

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 blocks 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.

3.9.16 Why Does Data Query or Loading Fail and "org.apache.carbondata.core.memory.MemoryException: Not enough memory" Is Displayed?

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

3.9.17 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.

4 Using ClickHouse

4.1 ClickHouse Overview

ClickHouse Table Engine

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 engines, which are the most important and frequently used ClickHouse table engines.

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 with materialized views, which are query views of other data tables at the upper layer. 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:

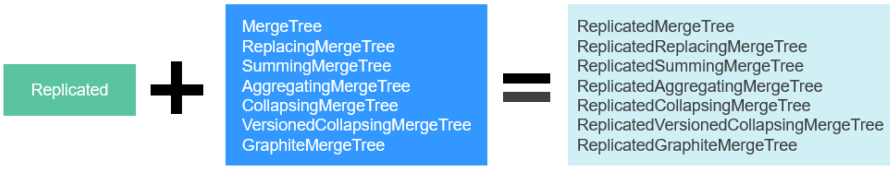
```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    Path String,
    Time DateTime,
    Value <Numeric_type>,
    Version <Numeric_type>
    ...
) ENGINE = GraphiteMergeTree(config_section)
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

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***.

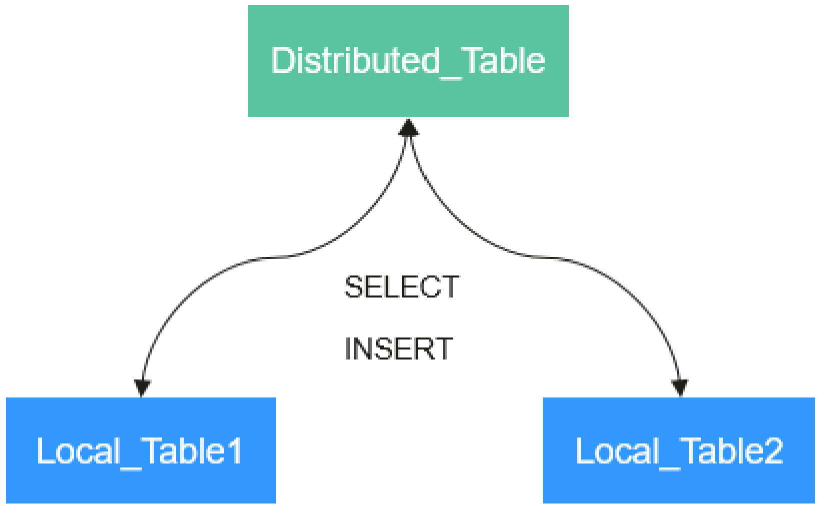
  – *Replica name*: Generally, **{replica}** is used.

  For details about the example, see **Creating a ClickHouse Table**.

- **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 4-1 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 4-1 describes the data types used in the ClickHouse component of MRS.

For details about the ClickHouse data types, see **ClickHouse Data Types**.

Table 4-1 ClickHouse data types

Category	Keyword	Type	Description
Type	Int8	Int8	Value range: [–128, 127]
	Int16	Int16	Value range: [–32768, 32767]
	Int32	Int32	Value range: [–2147483648, 2147483647]
	Int64	Int64	Value range: [–9223372036854775808, 9223372036854775807]
Floating point	Float32	Float	Similar to the Float type in C, a single-precision floating point number occupies 4 bytes in storage of the machine 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	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	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 DBMSs.
	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
Date and time	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	Viewing the 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	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	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	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.
Enumeration	Enum8	Enum8	Value range: [–128, 127] An Enum value stores the mapping of 'string'= integer, for example, **Enum8('hello' = 1, 'world' = 2)**.
	Enum16	Enum16	Value range: [–32768, 32767]
Nullable	Nullable	Nullable	Unless otherwise specified in the ClickHouse server configuration, NULL is the default value for any Nullable type. 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, ...)**

4.2 ClickHouse User Permission Management

4.2.1 ClickHouse User and Permission Management

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 4-2** lists the resources supported by ClickHouse permission management.

Resource permissions: **Table 4-3** lists the resource permissions supported by ClickHouse.

NOTE

This section does not support MRS 3.1.0 clusters in common mode (Kerberos authentication disabled).

Table 4-2 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 4-3 Resource permission list

Resource	Available Permission	Remarks
Database	CREATE	CREATE DATABASE/TABLE/VIEW/ DICTIONARY
Table/View	SELECT/INSERT	-

Prerequisites

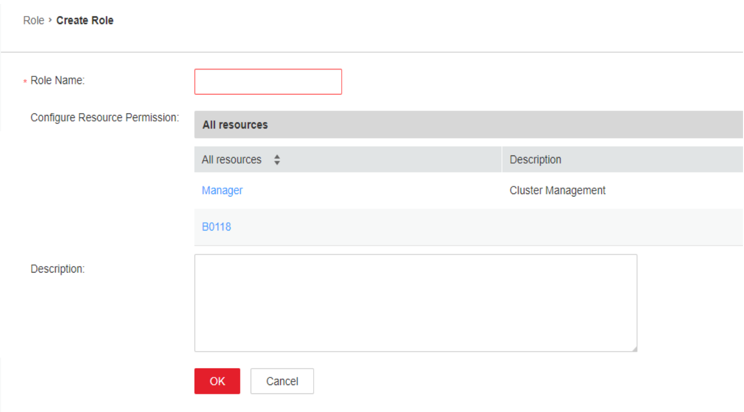
- The ClickHouse and Zookeeper services are running properly.
- The **ON CLUSTER** statement has been used to create a database or table in the cluster to ensure that the metadata of the database and table on each ClickHouse node is the same.

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.

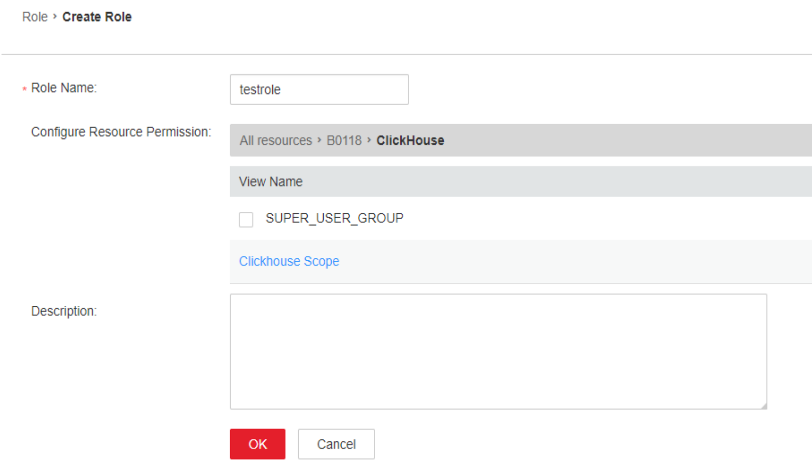
Determine whether to create a role with the ClickHouse administrator permissions based on service requirements.

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.

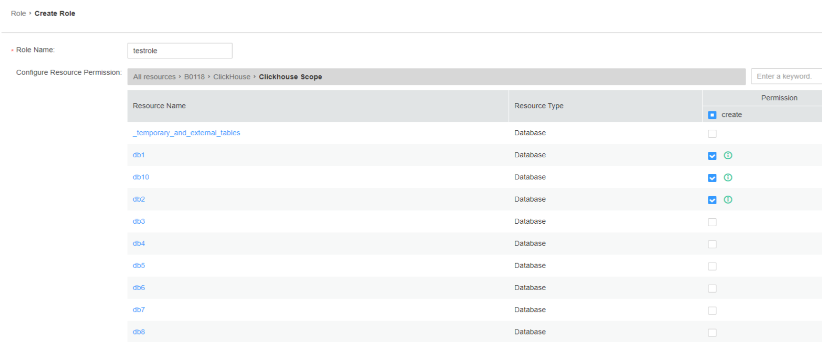
- If yes, go to **Step 3**.

- If no, go to **Step 4**.



Step 3 Select **SUPER_USER_GROUP** and click **OK**.

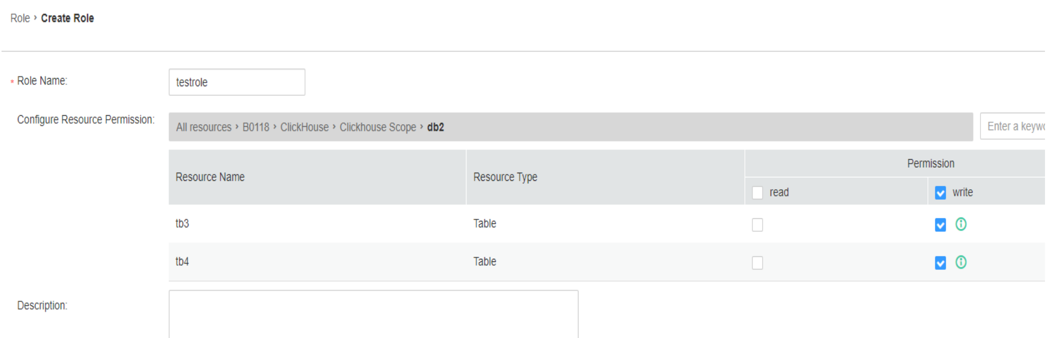
Step 4 Click **Clickhouse Scope**. The ClickHouse database resource list is displayed. If you select **create**, the role has the create permission on the database.



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

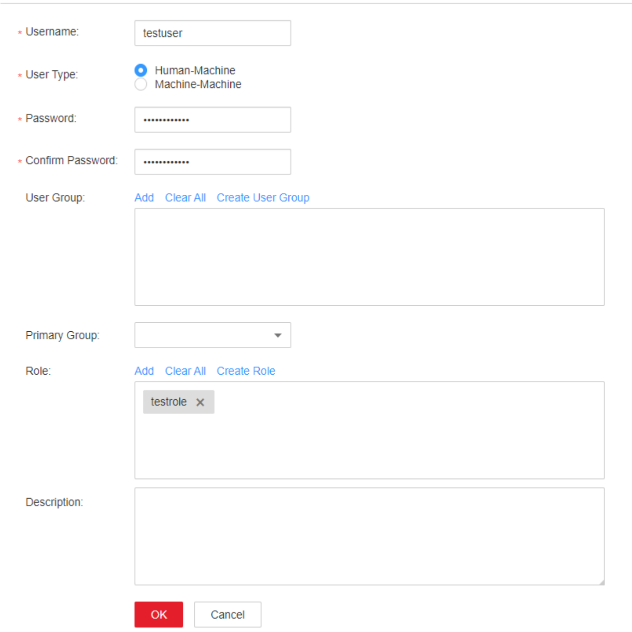
Step 1 Log in to Manager and choose **System** > **Permission** > **User** and click **Create**.

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.

  For an MRS 3.1.0 cluster, run the **export CLICKHOUSE_SECURITY_ENABLED=true** command first.

  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

For a cluster in normal mode, the username is **default**. Alternatively, you can create an administrator using the open source ClickHouse web UI. 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

To perform the following operations, you need to obtain the required roles, database or table names, IP addresses, and system domain names of the node where the corresponding ClickHouseServer instance is deployed.

- You can log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse** > **Instance** to obtain the service IP address of the ClickHouseServer instance.
- 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

For an MRS 3.1.0 cluster with Kerberos authentication enabled, additionally run the following command:

export CLICKHOUSE_SECURITY_ENABLED=true

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

4.2.2 Interconnecting ClickHouse With OpenLDAP for Authentication

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 4-2**.

Table 4-4 OpenLDAP parameters

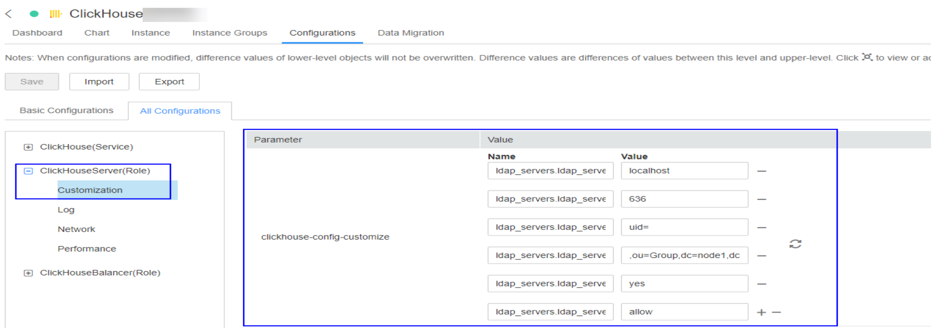
<table>
<tr><th>Parameter</th><th>Description</th><th>Example Value</th></tr>
<tr><td>ldap_servers.ldap_server_name.host</td><td>OpenLDAP server host name or IP address. This parameter cannot be empty.</td><td>localhost</td></tr>
<tr><td>ldap_servers.ldap_server_name.port</td><td>OpenLDAP server port number.
If enable_tls is set to true, the default port number is 636. Otherwise, the default port number is 389.</td><td>636</td></tr>
<tr><td>ldap_servers.ldap_server_name.auth_dn_prefix</td><td rowspan="2">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.</td><td>uid=</td></tr>
<tr><td>ldap_servers.ldap_server_name.auth_dn_suffix</td><td>,ou=Group,dc=node1,dc=com</td></tr>
<tr><td>ldap_servers.ldap_server_name.enable_tls</td><td>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.</td><td>yes</td></tr>
</table>

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 4-2 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 and cannot be empty.

- **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**.

  Note that you should use a comma (,) as the first non-space character of **auth_dn_suffix**.

- **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 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**

  Allowed encryption suite.

4.3 Using the ClickHouse Client

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.

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 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 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 current cluster, run the following command to authenticate the user. The current user must have the permission to create ClickHouse tables. For details about how to configure the permission, see **ClickHouse User and Permission Management**, and bind roles to the user. If Kerberos authentication is disabled for the current cluster, skip this step.

For an MRS 3.1.0 cluster, run the **export CLICKHOUSE_SECURITY_ENABLED=true** command first.

kinit *Component service user*

Example: **kinit clickhouseuser**

Step 6 Run the client command of the ClickHouse component.

Run the **clickhouse -h** command to view the command help of ClickHouse.

The command output is as follows:

```
Use one of the following commands:
clickhouse local [args]
clickhouse client [args]
clickhouse benchmark [args]
clickhouse server [args]
clickhouse performance-test [args]
clickhouse extract-from-config [args]
clickhouse compressor [args]
clickhouse format [args]
clickhouse copier [args]
clickhouse obfuscator [args]
...
```

For MRS 3.1.0, run the **clickhouse client** command to connect to the ClickHouse server.

- Command for using a non-SSL mode to log in to a ClickHouse cluster with Kerberos authentication disabled

  clickhouse client --host *IP address of the ClickHouse instance* **--port 9000 --user** *Username* **--password**

Enter the user password.

- Using SSL for login when Kerberos authentication is enabled for the current cluster:

  There are no default users in clusters with Kerberos authentication enabled. You must create a user on FusionInsight Manager. For details about how to create a user, see **ClickHouse User and Permission Management**.

  After the user authentication is successful, you do not need to carry the **--user** and **--password** parameters when logging in to the client as the authenticated user.

  clickhouse client --host *IP address of the ClickHouse instance* **--port 9440 --secure**

For MRS 3.1.2 or later, run the **clickhouse client** command to connect to the ClickHouse server.

- Command for using a non-SSL mode to log in to a ClickHouse cluster with Kerberos authentication disabled

  clickhouse client --host *IP address of the ClickHouse instance* **--port 9000 --user** *Username* **--password**

  Enter the user password.

- Using SSL for login when Kerberos authentication is enabled for the current cluster:

  There are no default users in clusters with Kerberos authentication enabled. You must create a user on FusionInsight Manager. For details about how to create a user, see **ClickHouse User and Permission Management**.

  clickhouse client --host *IP address of the ClickHouse instance* **--port 9440 --user** *Username* **--password --secure**

  Enter the user password.

Run the **quit;** command to exit the ClickHouse server connection.

Table 4-5 describes related parameters.

Table 4-5 Parameters of the **clickhouse client** command

Parameter	Description
--host	Host name of the server. The default value is **localhost**. You can use the host name or IP address of the node where the ClickHouse instance is located. **NOTE** You can log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse** > **Instance** to obtain the service IP address of the ClickHouseServer instance.

Parameter	Description
--port	Port for connection. ● If the SSL security connection is used, the default port number is **9440**, the parameter **--secure** must be carried. For details about the port number, search for the **tcp_port_secure** parameter in the ClickHouseServer instance configuration. ● If non-SSL security connection is used, the default port number is **9000**, the parameter **--secure** does not need to be carried. For details about the port number, search for the **tcp_port** parameter in the ClickHouseServer instance configuration.
--user	Username. You can create a user on FusionInsight Manager and bind roles to it. For details about how to create a user, see **ClickHouse User and Permission Management**. ● If Kerberos authentication has been enabled for the current cluster (the cluster is in security mode) and the user authentication is successful, you do not need to carry the **--user** and **--password** parameters during your login to the client as the authenticated user. You must create a user with this name on Manager because there is no default user in the Kerberos cluster scenario. ● If Kerberos authentication has not been enabled for the current cluster (the cluster is in normal mode), you cannot use the ClickHouse user created on FusionInsight Manager if you need to specify the username and password when you log in to the client. You need to execute the **create user SQL** statement on the client to create a ClickHouse user. If you do not need to specify the username and password during your login to the client, the default user is used by default.
--password	Password. The default password is an empty string. This parameter is used together with the **--user** parameter. You can set a password when creating a user on Manager.
--query	Query to process when using non-interactive mode.
--database	Current default database. The default value is **default**, which is the default configuration on the server.
--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.

Parameter	Description
--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**.

----End

4.4 Creating a ClickHouse Table

ClickHouse implements the replicated table mechanism based on the ReplicatedMergeTree engine and ZooKeeper. When creating a table, you can specify an engine to determine whether the table is highly available. Shards and replicas of each table are independent of each other.

ClickHouse also implements the distributed table mechanism based on the Distributed engine. Views are created on all shards (local tables) for distributed query, which is easy to use. ClickHouse has the concept of data sharding, which is one of the features of distributed storage. That is, parallel read and write are used to improve efficiency.

The ClickHouse cluster table engine that uses Kunpeng as the CPU architecture does not support HDFS and Kafka.

Viewing cluster and Other Environment Parameters of ClickHouse

Step 1 Use the ClickHouse client to connect to the ClickHouse server by referring to **Using the ClickHouse Client**.

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 Log in to the ClickHouse node using the client, for example, **clickhouse client --host** *node-master3QsRI* **--multiline --port 9440 --secure;**

NOTE

node-master3QsRI is the value of **host_name** obtained in **Step 2** in **Viewing cluster and Other Environment Parameters of ClickHouse**.

Step 2 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, run the following commands to create a ReplicatedMergeTree table named **test** on the **default_cluster_1** node and in the **default** database:

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;

The parameters are described as follows:

- The **ON CLUSTER** syntax indicates the distributed DDL, that is, the same local table can be created on all instances in the cluster after the statement is executed once.
- **default_cluster_1** is the cluster identifier obtained in **Step 2** in **Viewing cluster and Other Environment Parameters of ClickHouse**.

> ⚠ 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 3 Create a distributed table using the Distributed engine.

For example, run the following commands to create a distributed table named **test_all** on the **default_cluster_1** node and in the **default** database:

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_active─┐
│ 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 obtained in **Step 2** in **Viewing cluster and Other Environment Parameters of ClickHouse**.
- **default** indicates the name of the database where the local table is located.
- **test** indicates the name of the local table. In this example, it is the name of the table created in **Step 2**.
- (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 Log in to the ClickHouse node on the client. Example:

clickhouse client --host *node-master3QsRI* **--multiline --port 9440 --secure;**

NOTE

node-master3QsRI is the value of **host_name** obtained in **Step 2** in **Viewing cluster and Other Environment Parameters of ClickHouse**.

Step 2 After creating a table by referring to **Creating a Local Replicated Table and a distributed Table**, you can insert data to the local table.

For example, run the following command to insert data to the local table **test**:

insert into test values(toDateTime(now()), rand());

Step 3 Query the local table information.

For example, run the following command to query data information of the table **test** in **Step 2**:

select * from test;

```
SELECT *
FROM test

┌──────────EventDate─┬─────────id─┐
│ 2020-11-05 21:10:42 │ 1596238076 │
└─────────────────────┴────────────┘

1 rows in set. Elapsed: 0.002 sec.
```

Step 4 Query the distributed table.

For example, the distributed table **test_all** is created based on table **test** in **Step 3**. Therefore, the same data in table **test** can also be queried in table **test_all**.

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 5 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;**

NOTE

The **shard_num** values of **node-group-1tXED0003** and **node-master3QsRI** are the same by performing **Step 2**.

show tables;

```
SHOW TABLES

┌─name─────┐
│ test     │
│ test_all │
└──────────┘
```

Step 6 Query the local table data. For example, run the following command to query data in table **test** 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 7 Switch to the shard node with different **shard_num** value and query the data of the created table.

For example, run the following command to exit the **node-group-1tXED0003** node:

exit;

Switch to the **node-group-1tXED0001** node. The **shard_num** values of **node-group-1tXED0001** and **node-master3QsRI** are different by performing **Step 2**.

clickhouse client --host *node-group-1tXED0001* **--multiline --port 9440 --secure;**

Query the local table **test**. Data cannot be queried on the different shard node because table **test** is a local table.

select * from test;

```
SELECT *
FROM test

Ok.
```

Query data in the distributed table **test_all**. The data can be queried properly.

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

4.5 ClickHouse Data Import

4.5.1 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. 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.

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 **ClickHouse User and Permission Management**. 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

4.5.2 Interconnecting ClickHouse with OBS

Using S3 Table Functions

Step 1 Log in to the active OMS node.

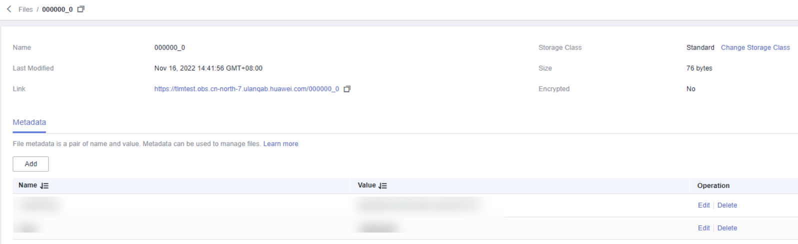
Step 2 Run the following command to obtain the data stored in OBS:

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 4-3 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

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?**

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

After the modification, click **Save**.

4.5.3 Synchronizing Kafka Data to ClickHouse

This section describes how to create a Kafka 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 4-6 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:

Parameter	Mandatory	Description
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.

   For an MRS 3.1.0 cluster, run the **export CLICKHOUSE_SECURITY_ENABLED=true** command first.

   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:
  – For versions earlier than MRS 3.x, click the cluster name to go to the cluster details page and choose **Components** > **ZooKeeper** > **Instances**. View the IP addresses of the ZooKeeper role instances.
  – For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > *Name of the desired cluster* > **Services** > **ZooKeeper** > **Instance**. View the IP addresses of the ZooKeeper role instances.
- **--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 **Using the ClickHouse Client**.

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 **ClickHouse User and Permission Management**. 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;
```

```
ClickHouseKXya.mrs-2xxk.com :) select * from kafka_dest_tbl3;

SELECT *
FROM kafka_dest_tbl3

Query id: 386ed9db-26b6-4823-8ce5-5b2de14384bb

┌─id─┬─age─┬─msg──────────┐
│ 31 │  30 │ 31 years old │
└────┴─────┴──────────────┘
┌─id─┬─age─┬─msg──────────┐
│ 32 │  30 │ 31 years old │
└────┴─────┴──────────────┘
┌─id─┬─age─┬─msg──────────┐
│ 33 │  30 │ 31 years old │
└────┴─────┴──────────────┘
┌─id─┬─age─┬─msg──────────┐
│ 35 │  30 │ 31 years old │
└────┴─────┴──────────────┘

4 rows in set. Elapsed: 0.003 sec.
```

----End

4.5.4 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 GaussDB(DWS) service 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 GaussDB(DWS) cluster and obtained the username and password of the database where the related table is located.

- The MRS 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.

Importing GaussDB(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 GaussDB(DWS) database using the username and password of the database in the GaussDB(DWS) cluster. For details, see **Adding a Connection**.

Step 3 Export table data from the GaussDB(DWS) database to a CSV file.

1. (Optional) If a table and data already exist in the GaussDB(DWS) database, skip this step. The following demonstrates how to create a test table in GaussDB(DWS) and insert test data.

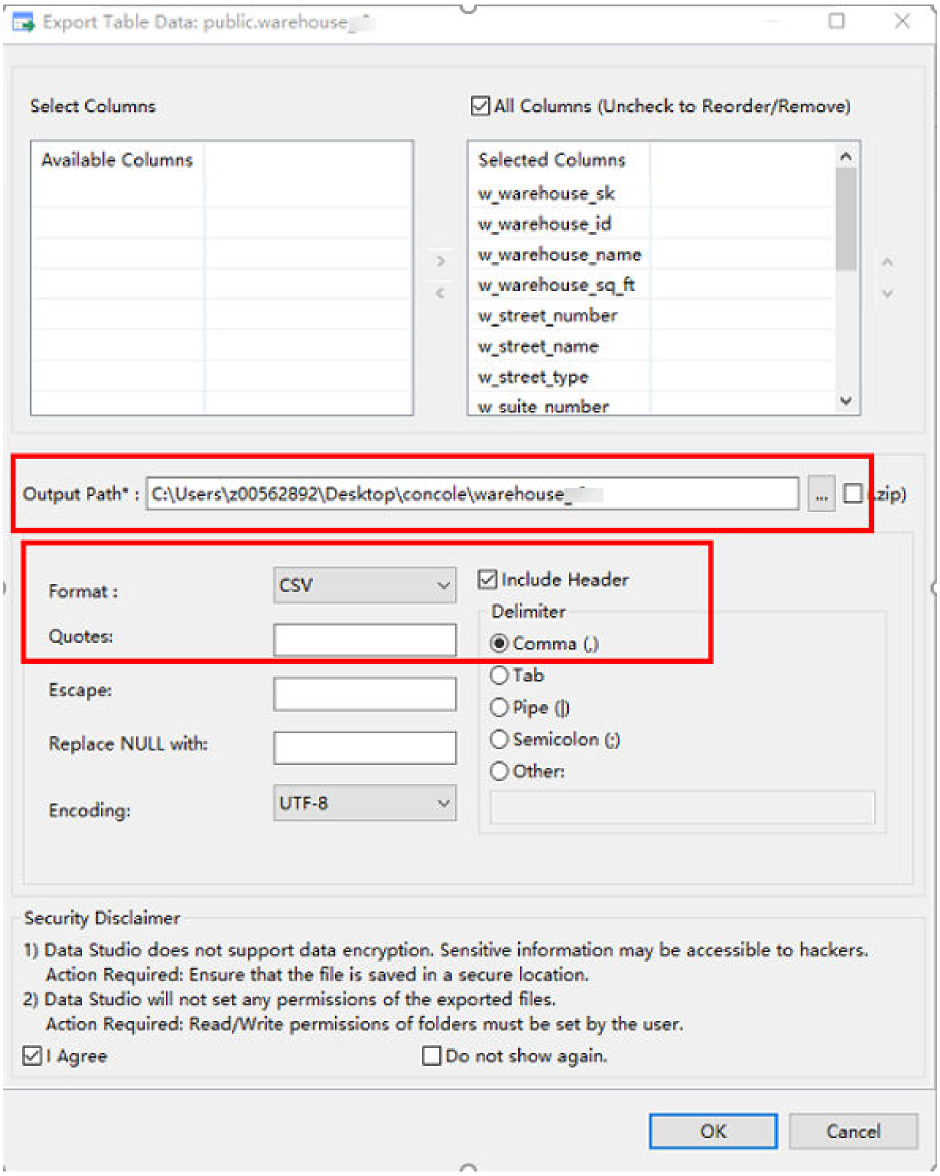
   Use Data Studio to create a test table **warehouse_t1** and insert test data.

```
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 GaussDB(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 **ClickHouse User and Permission Management**. 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 GaussDB(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

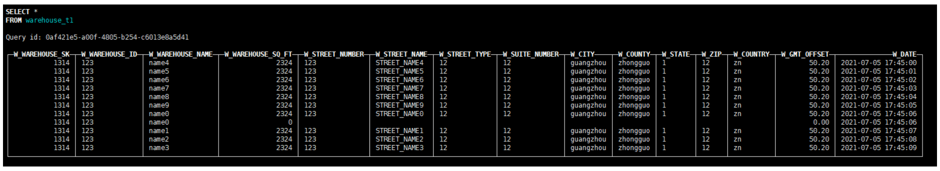
Step 13 After the import is complete, log in to the ClickHouse client, connect to the ClickHouse instance node where the data is imported, and run the query command to check the import result.

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;



```
SELECT *
FROM warehouse_t1

Query id: 0af421e5-a00f-4805-b254-c6013e8a5d41
```

W_WAREHOUSE_SK	W_WAREHOUSE_ID	W_WAREHOUSE_NAME	W_WAREHOUSE_SQ_FT	W_STREET_NUMBER	W_STREET_NAME	W_STREET_TYPE	W_SUITE_NUMBER	W_CITY	W_COUNTY	W_STATE	W_ZIP	W_COUNTRY	W_GMT_OFFSET	W_DATE
1314	123	name4	2324	123	STREET_NAME4	12	12	guangzhou	zhongguo	1	12	zn	50.20	2021-07-05 17:45:00
1314	123	name5	2324	123	STREET_NAME5	12	12	guangzhou	zhongguo	1	12	zn	50.20	2021-07-05 17:45:01
1314	123	name6	2324	123	STREET_NAME6	12	12	guangzhou	zhongguo	1	12	zn	50.20	2021-07-05 17:45:02
1314	123	name7	2324	123	STREET_NAME7	12	12	guangzhou	zhongguo	1	12	zn	50.20	2021-07-05 17:45:03
1314	123	name8	2324	123	STREET_NAME8	12	12	guangzhou	zhongguo	1	12	zn	50.20	2021-07-05 17:45:04
1314	123	name9	2324	123	STREET_NAME9	12	12	guangzhou	zhongguo	1	12	zn	50.20	2021-07-05 17:45:05
1314	123	name0	2324	123	STREET_NAME0	12	12	guangzhou	zhongguo	1	12	zn	50.20	2021-07-05 17:45:06
1314	123	name0	0										0.00	2021-07-05 17:45:06
1314	123	name1	2324	123	STREET_NAME1	12	12	guangzhou	zhongguo	1	12	zn	50.20	2021-07-05 17:45:07
1314	123	name2	2324	123	STREET_NAME2	12	12	guangzhou	zhongguo	1	12	zn	50.20	2021-07-05 17:45:08
1314	123	name3	2324	123	STREET_NAME3	12	12	guangzhou	zhongguo	1	12	zn	50.20	2021-07-05 17:45:09

----End

4.5.5 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
```


4.6 Enterprise-Class Enhancements of ClickHouse

4.6.1 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 4-4** 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 4-4 Accessing ClickHouse nodes through ELB

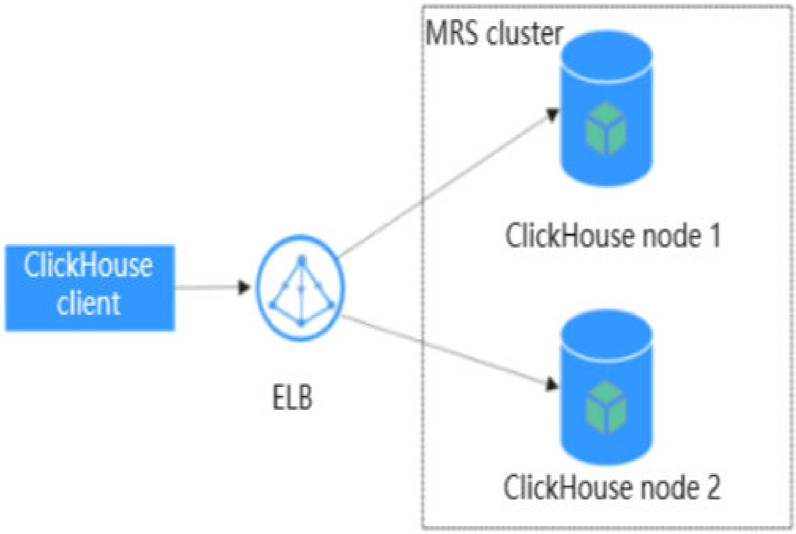


Table 4-7 lists the advantages of the ELB-based deployment over BalancedClickhouseDataSource.

Table 4-7 Differences between ELB-based deployment and BalancedClickhouseDataSource

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.
BalancedClickhouseDataSource	• Causes load imbalance due to random allocation of requests. • Lacks of sufficient fault detection capabilities.

Table 4-8 lists the supported protocols and ports for accessing ClickHouse through ELB. Configure them as required.

Table 4-8 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.
- The MRS 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.

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.

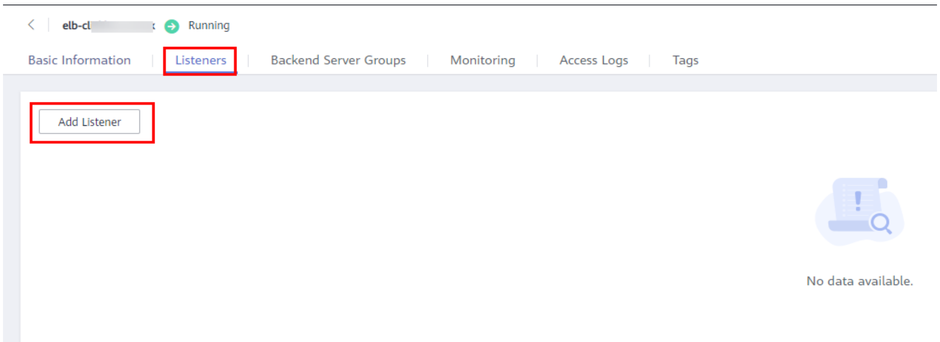
----End

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.

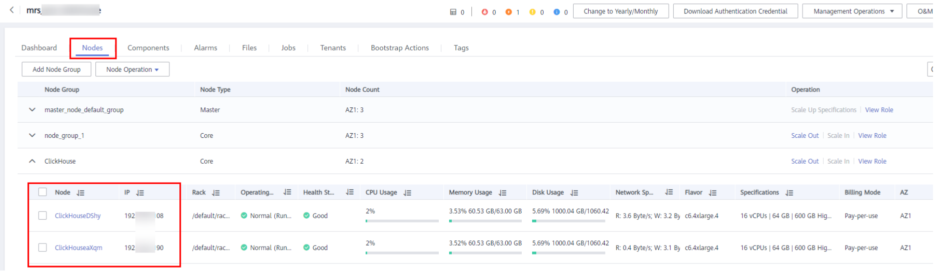
Set **Load Balancing Algorithm** to **Weighted round robin** and click **Finish**. On the displayed page, click **OK**.

----End

Adding ClickHouse backend servers

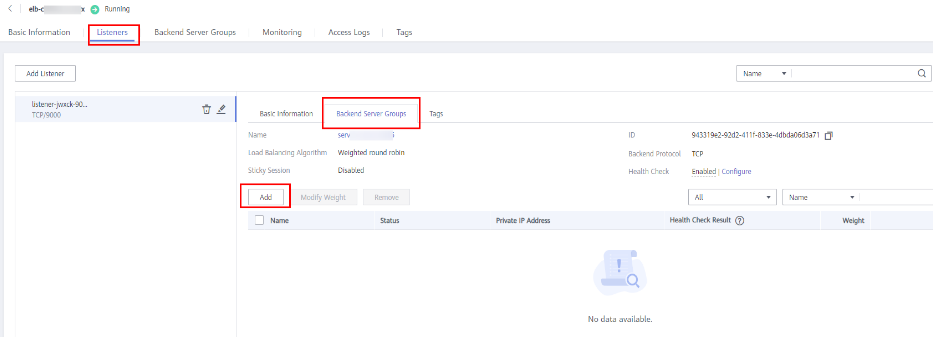
Step 1 Switch to the MRS console and click the MRS cluster to be interconnected.

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**.

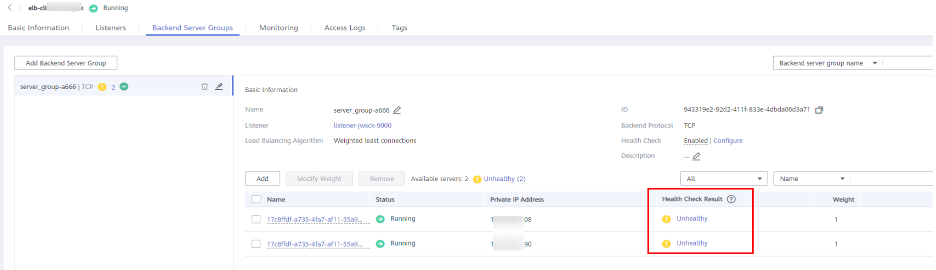
Step 6 On the displayed page, set **Batch Add Ports** to **9000** and click **OK**. Confirm your configurations and click **Finish**.

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**.

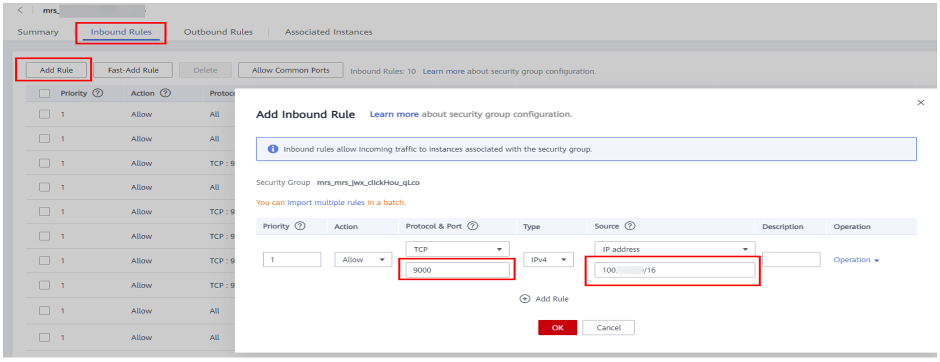


To solve this issue, you need to configure the inbound rule of the security group for the ClickHouse backend server to allow access from 100.125.0.0/16. The procedure is as follows:

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 **Using the ClickHouse Client**. Note that the **host** parameter in the **clickhouse client** command must be set to the private IP address of ELB obtained in **Step 4**.

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

4.6.2 Enabling the mysql_port Configuration for ClickHouse

Use the MySQL client to connect to ClickHouse.

Procedure

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.

After the modification, click **Save**.

Step 2 Click the **Dashboard** tab, click **More**, and select **Restart Instance**; alternatively, click **More** and select **Instance Rolling Restart**.

----End

4.7 ClickHouse Performance Tuning

4.7.1 Solution to the "Too many parts" Error in Data Tables

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 Merge Operations**.

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**, locate the **clickhouse-config-customize** parameter, add the configuration in **Table 4-9**, 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 4-9 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.

4.7.2 Accelerating Merge Operations

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.

4.7.3 Accelerating 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.

4.8 ClickHouse O&M Management

4.8.1 ClickHouse Log Overview

Log Description

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 automatically 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 4-10 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

Log Type	Log File Name	Description
	/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
	/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 4-11 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 4-11 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 4-12 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

4.8.2 ClickHouse Cluster Management

4.8.2.1 ClickHouse Cluster Configuration

Background

ClickHouse uses the multi-shard and multi-replica deployment architecture to implement the cluster high availability. Multiple shards are defined in each cluster, and each shard has two or more replicas. If a node is faulty, replicas on other nodes in the shard can take over services from the faulty node, ensuring service continuity and improving cluster stability.

NOTE

This section applies only to MRS 3.1.0.

Cluster Configuration

Step 1 Log in to Manager and choose **Cluster** > **Services** > **ClickHouse** > **Configurations** > **All Configurations**.

Step 2 Add the custom configuration items in **Table 4-13** to the **clickhouse-metrika-customize** parameter.

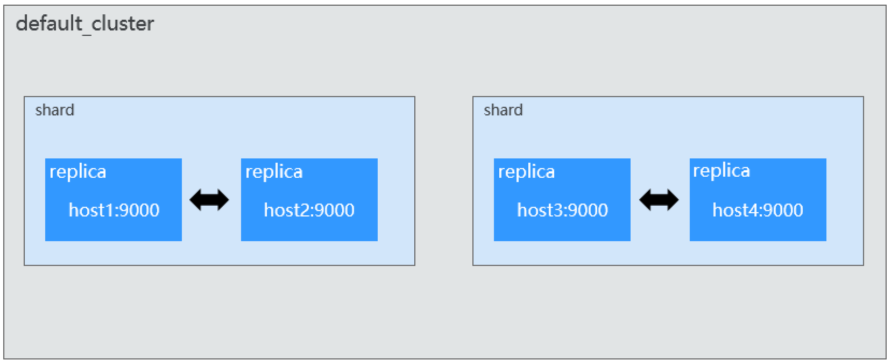
Table 4-13 Custom parameters

Parameter	Value
clickhouse_remote_servers.example_cluster.shard[1].replica[1].host	host1.9bf17e66-e7ed-4f21-9dfc-34575f955ae6.com
clickhouse_remote_servers.example_cluster.shard[1].replica[1].port	9000
clickhouse_remote_servers.example_cluster.shard[1].replica[2].host	host2.9bf17e66-e7ed-4f21-9dfc-34575f955ae6.com
clickhouse_remote_servers.example_cluster.shard[1].replica[2].port	9000
clickhouse_remote_servers.example_cluster.shard[1].internal_replication	true
clickhouse_remote_servers.example_cluster.shard[2].replica[1].host	host3.9bf17e66-e7ed-4f21-9dfc-34575f955ae6.com
clickhouse_remote_servers.example_cluster.shard[2].replica[1].port	9000
clickhouse_remote_servers.example_cluster.shard[2].replica[2].host	host4.9bf17e66-e7ed-4f21-9dfc-34575f955ae6.com
clickhouse_remote_servers.example_cluster.shard[2].replica[2].port	9000
clickhouse_remote_servers.example_cluster.shard[2].internal_replication	true

Step 3 Click **Save**.

----End

The following figure shows the cluster architecture:



The following describes the parameters:

- **default_cluster**
  - **default_cluster** indicates the name of the current cluster.
  - The current cluster has two shards. Each shard has two replicas, and each replica corresponds to a ClickHouse instance node.
  - **internal_replication** indicates whether internal replication is performed between replicas. It takes effect when data is inserted into shards through the cluster.

    The default value is **true**, indicating that data is written to only one replica. (Data is synchronized between replica through replicated tables to ensure data consistency.)

    If this parameter is set to **false** (**not recommended**), same data is written to all replicas of the shard. (Data between replicas is not strongly consistent, and full synchronization cannot be ensured.)
- **macros**

  macros indicates IDs the shard and replica where the current instance node resides. It can be used to distinguish different replicas.

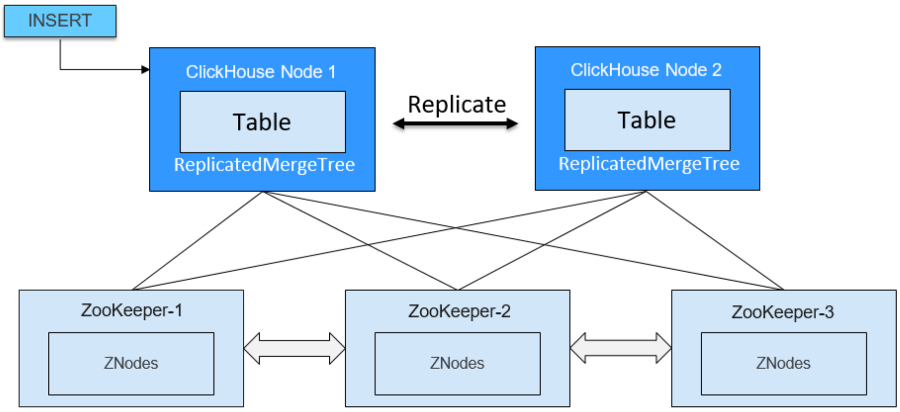
  For example, the preceding example shows the configuration of the **host3** instance. The shard ID of the instance is **2** and the replica ID is **1**.

This section describes how to configure sharding and replication. For details about how to synchronize data between replicas in the ClickHouse cluster, see **Replication**.

Replication

ClickHouse uses ZooKeeper and the ReplicatedMergeTree engine (of Replicated series) to implement replication. Replication uses a multi-master scheme. The INSERT statement can be sent to any replica, and data is replicated to other replicas in the shard asynchronously.

In the following figure, Node 1 and Node 2 correspond to **host1** and **host2** in **Cluster Configuration**.



After the ClickHouse cluster is successfully created, three ZooKeeper nodes are created by default. ZooKeeper stores the metadata of the ClickHouse table during replication.

For details about the ZooKeeper node information, see the **config.xml** file in the **${BIGDATA_HOME}/FusionInsight_ClickHouse_***Version number***/***x_x_ClickHouse instance name***/etc** directory.

```
<yandex>
 ...
 <zookeeper>
   <node index="1">
     <host>node-master1lrgj.9bf17e66-e7ed-4f21-9dfc-34575f955ae6.com</host>
     <port>2181</port>
   </node>
   <node index="2">
     <port>2181</port>
     <host>node-master2vocd.9bf17e66-e7ed-4f21-9dfc-34575f955ae6.com</host>
   </node>
   <node index="3">
     <host>node-master3xwmu.9bf17e66-e7ed-4f21-9dfc-34575f955ae6.com</host>
     <port>2181</port>
   </node>
 </zookeeper>
 ...
```

For details about how to use the cluster after configuration, see **Creating a ClickHouse Table**.

4.8.2.2 Expanding the Data Disk Capacity of a 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.

This section applies only to MRS 3.1.0.

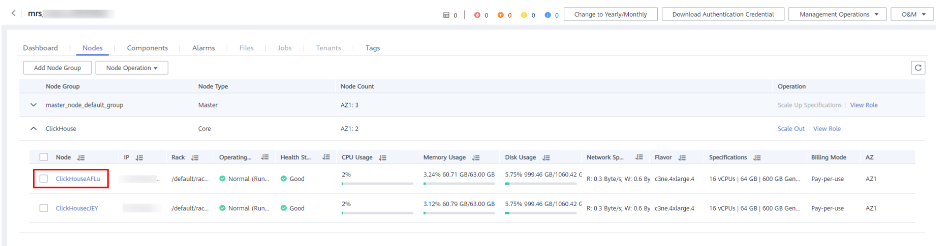
Prerequisites

- 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.

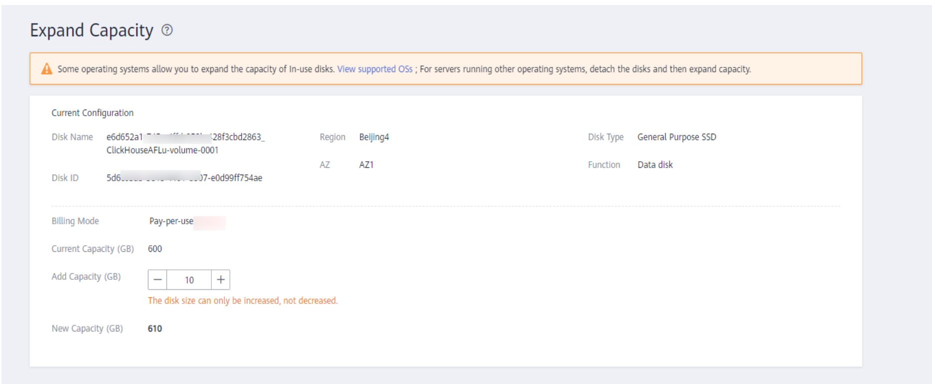


Step 3 In the row of the target data disk, click **Expand Capacity**.

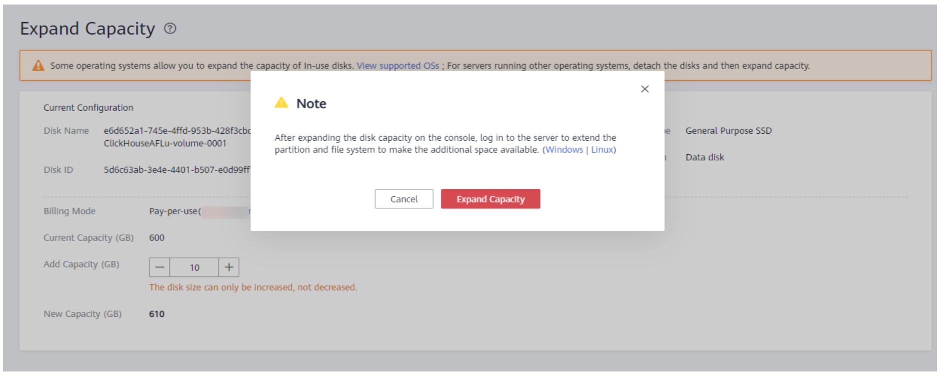
NOTE

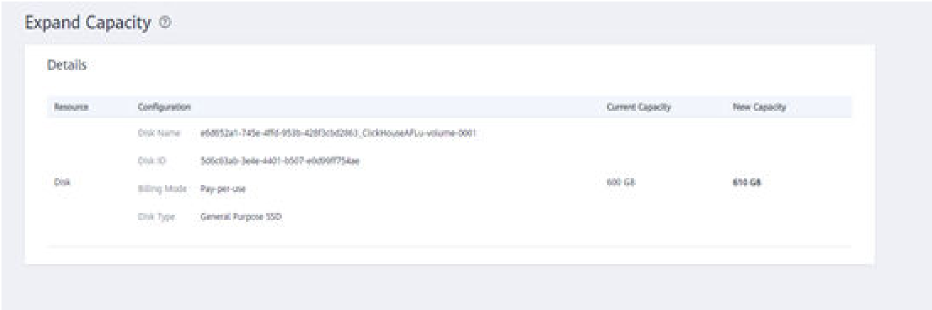
If only the system disk is displayed on the current page and no data disk exists, the data disk of the ClickHouse node cannot be expanded using this method.

Step 4 Modify the disk capacity to be added in **Add Capacity (GB)** and click **Next**.

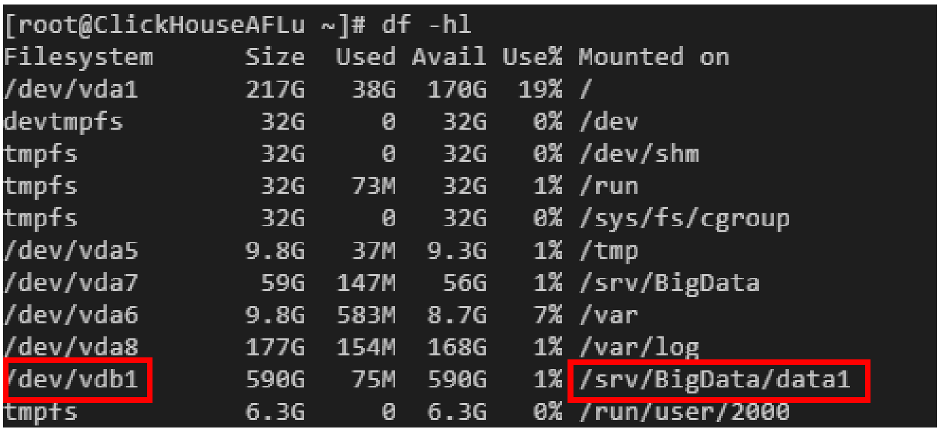


Step 5 Read the note for capacity expansion carefully, click **Expand Capacity**, confirm the information about the expanded disk capacity, and click **Submit**.





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 mounting for partitions at system start by modifying the **/etc/fstab** file.

Step 11 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. 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, add **/srv/BigData/data2/clickhouse/** to **/srv/BigData/data1/clickhouse/**, and the new directory is **/srv/BigData/data1/clickhouse/,/srv/BigData/data2/clickhouse/**.

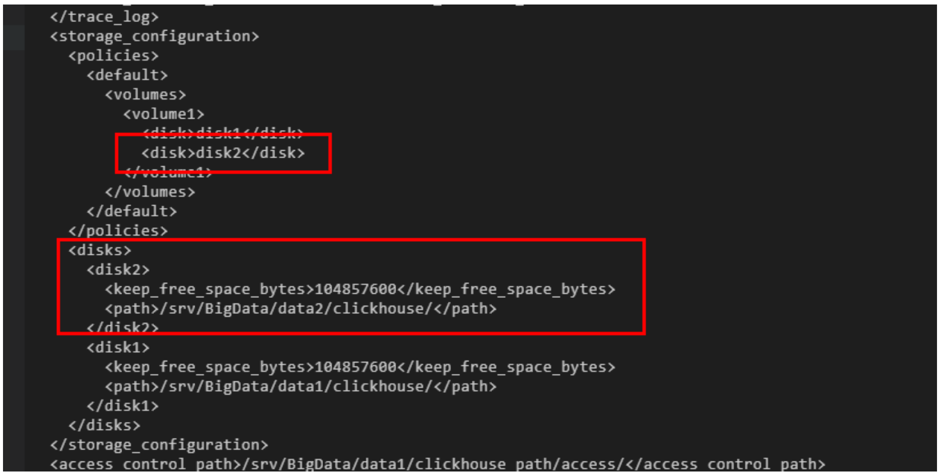
Step 13 After the new directory is added, click **Save** to save the configuration. Click **Dashboard**, choose **More** > **Synchronize Configuration**, and click **OK**.

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

4.8.3 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

4.8.4 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

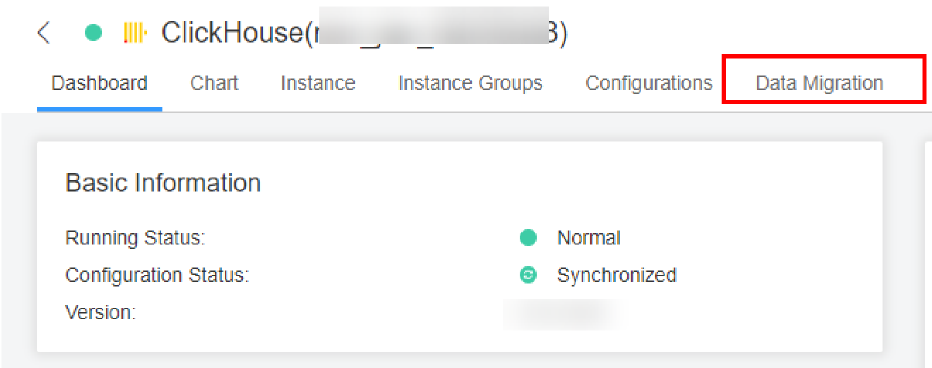
- The ClickHouse and Zookeeper services are running properly. The ClickHouseServer instances on the source and destination nodes are normal.
- The destination node has the data table to be migrated and the table is a partitioned MergeTree table.
- Before creating a migration task, ensure that all tasks 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.

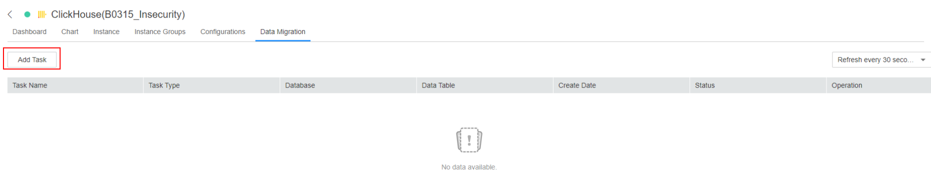
- The ClickHouse data directory on the destination node has sufficient space.

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 4-14**.

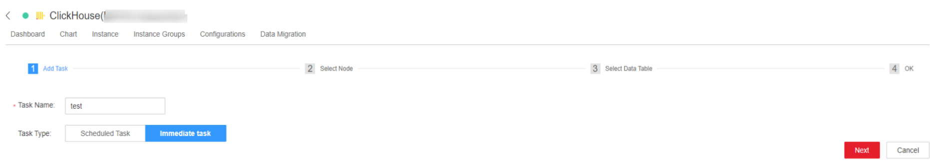
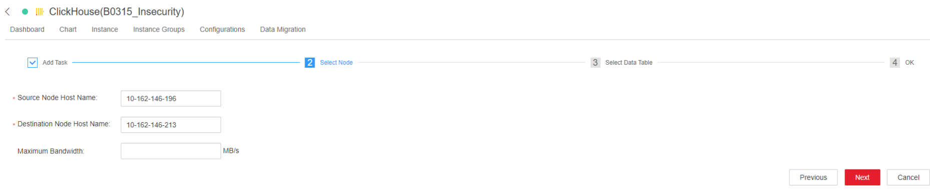


Table 4-14 Migration task parameters

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.

Parameter	Description
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**.

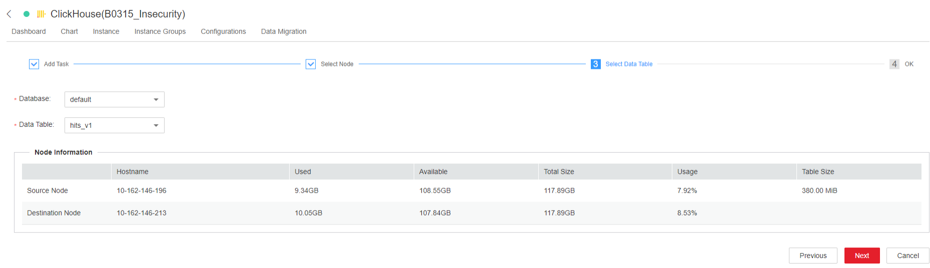


NOTE

- 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.

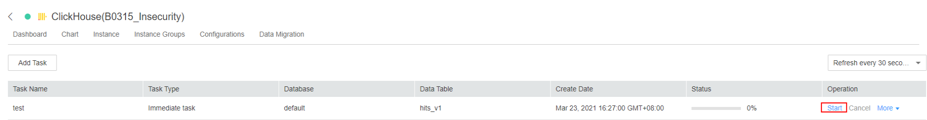
Step 5 On the **Select Data Table** page, click next to **Database**, select the database to be migrated on the source node, and select the data table to be migrated for **Data Table**. The data table drop-down list displays the partitioned MergeTree tables in the selected database. In the **Node Information** 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

4.9 Common ClickHouse SQL Syntax

4.9.1 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    │
└────────┘
```

4.9.2 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 migrates data based on table partitions. If you do not use **PARTITION BY** in the table creation statement, the table data cannot be migrated as described 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_name*2 [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
```

4.9.3 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 │
└────┴──────┘
```

4.9.4 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;
```

4.9.5 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 │              │                │        │                │              │
└─────────┴────────┴──────────────┴────────────────┴────────┴────────────────┴──────────────┘
```

4.9.6 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.

4.9.7 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 │            │                  │         │                  │                │
└─────────┴───────┴─────────────┴──────────────────┴─────────┴──────────────────┴────────────────┘
```

4.9.8 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***;**.

4.9.9 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 │
└───────┘
```

4.10 ClickHouse FAQ

4.10.1 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.

4.10.2 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. 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.

   clickhouse --client --port 9002 --password *xxx* **-m --query='INSERT INTO default.student_hive FORMAT CSV' < /data/bigdata/hive/student.csv**

4.10.3 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.

4.10.4 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.
- 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 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

5 Using DBService

5.1 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 5-1 DBService log list

Type	Log File Name	Description
DBServer run log	dbservice_serviceCheck.log	Run log file of the service check script

Type	Log File Name	Description
	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
	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.)

Type	Log File Name	Description
	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
	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

Type	Log File Name	Description
DBService audit log	dbservice_audit.log	Audit log file that records DBService operations, such as backup and restoration operations

Log Format

The following table lists the DBService log formats.

Table 5-2 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.

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.

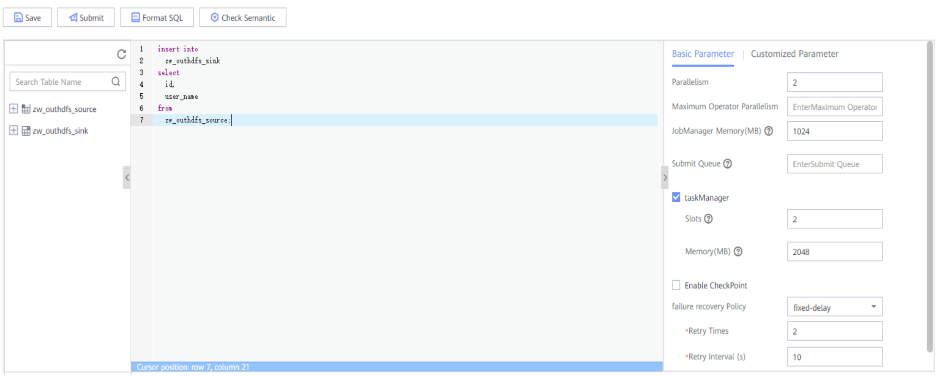
NOTE

This section applies to only MRS 3.1.0 or later.

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.



- 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

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 jobs, and multiple metrics can be combined into one job for computing. 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.

Item	Description
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 verify, parse, optimize, convert a job into a Flink job, and submit the job for running based on the entered SQL statements.
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 6-1 Flink web UI application process

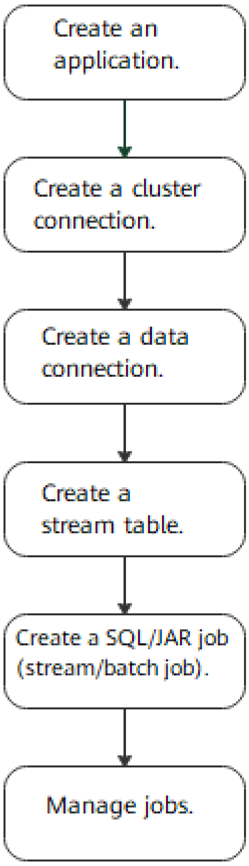


Table 6-2 Description of the Flink web UI application process

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.

  For details, see **Configuring Flink Authentication and Encryption**.

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, monitoring 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

  It applies to MRS 3.*x* or later.
  - 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

  It applies to MRS 3.*x* or later.
  - 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

  It applies to MRS 3.*x* or later.
  - 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

  It applies to MRS 3.*x* or later.
  - 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

  It applies to MRS 3.*x* or later.

  - 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:

  It applies to MRS 3.*x* or later.

  - 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	Default Value	Mandatory
jobmanager.web.403-redirect-url	Web page access error 403. If 403 error occurs, the page switch to a specified page.	-	Yes
jobmanager.web.404-redirect-url	Web page access error 404. If 404 error occurs, the page switch to a specified page.	-	Yes
jobmanager.web.415-redirect-url	Web page access error 415. If 415 error occurs, the page switch to a specified page.	-	Yes
jobmanager.web.500-redirect-url	Web page access error 500. If 500 error occurs, the page switch to a specified page.	-	Yes

- HTML5 security

  It applies to MRS 3.*x* or later.

  - 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.

NOTE

For versions earlier than MRS 3.x, only Kerberos authentication is supported.

- **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
```

NOTE

For versions earlier than MRS 3.x, set **security.kerberos.login.keytab** to **/home/demo/flink/release/keytab/flinkuser.keytab**.

- 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 number of the Kafka server.

- **SSL encryption**
  - Configure the server.

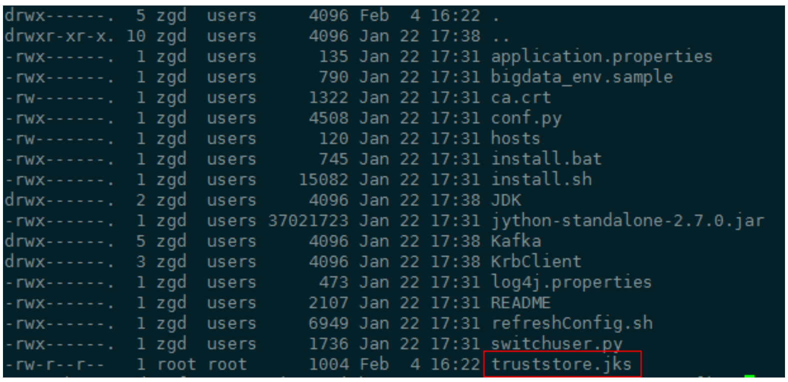
    Log in to FusionInsight Manager, choose **Cluster** > **Services** > **Kafka** > **Configurations**, and set **Type** to **All**. Search for **ssl.mode.enable** and set it to **true**.

  - Configure the client.
    i. Log in to FusionInsight Manager, choose **Cluster >** ***Name of the desired cluster*** **> Services > Kafka > More > Download Client** to download 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. 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.

       --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.
- For versions earlier than MRS 3.*x*, Flink does not support security cookie authentication.

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`

<table>
<tr><th>Authentication Mode</th><th>Description</th><th>Configuration Method</th></tr>
<tr><td>Security cookie authentication</td><td>-</td><td>
1. Generate the generate_keystore.sh script by referring to Example of Issuing a Certificate and place it in the bin directory of the Flink client. In the bin directory of the Flink client, run 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. 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.

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:
<code>security.cookie: ae70acc9-9795-4c48-ad35-8b5adc8071744f605d1d-2726-432e-88ae-dd39bfec40a9</code>

NOTE
Obtain the SSL certificate and save it to the Flink client. For details, see Example of Issuing a Certificate.

The validity period of the SSL certificate obtained by using the generate_keystore.sh script preset on the MRS client is 5 years. The validity period of the SSL certificate obtained by referring to Example of Issuing a Certificate is 10 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.
</td></tr>
</table>

Authentication Mode	Description	Configuration Method
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. Enable SSL and configure the SSL encryption algorithm. For MRS 3.x or later, see **Table 6-7**. Modify the parameters as required.

Table 6-7 Parameter description

Parameter	Example Value	Description
security.ssl.enabled	true	Enable SSL.

Parameter	Example Value	Description
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.

For versions earlier than MRS 3.x, see **Table 6-8**.

Table 6-8 Parameter description

Parameter	Example Value	Description
security.ssl.internal.enabled	true	Enable internal 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_RSA_WITH_AES128CBC_SHA256	Configure the SSL encryption algorithm.

For versions earlier than MRS 3.x, the following parameters in **Table 6-9** do not exist in the default Flink configuration of MRS. If you want to enable SSL for external connections, add the following parameters. After SSL for external connection is enabled, the native Flink page cannot be accessed using a Yarn proxy, because the Yarn open-source version cannot process HTTPS requests using a proxy. However, you can create a Windows VM in the same VPC of the cluster and access the native Flink page from the VM.

Table 6-9 Parameter description

Parameter	Example Value	Description
security.ssl.rest.enabled	true	Enable external SSL. If this parameter is set to **true**, set the related parameters by referring to **Table 6-9**.
security.ssl.rest.keystore	${path}/flink.keystore	Path for storing the **keystore**.
security.ssl.rest.keystore-password	-	A user-defined password of **keystore**.
security.ssl.rest.key-password	-	A user-defined password of the SSL key.
security.ssl.rest.truststore	${path}/flink.truststore	Path for storing the **truststore**.
security.ssl.rest.truststore-password	-	A user-defined password of **truststore**.

NOTE

- Enabling SSL for data transmission between TaskManagers may pose great impact on the system performance.
- 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.

2. Generate the **generate_keystore.sh** script by referring to **Example of Issuing a Certificate** and place it in the **bin** directory of the Flink client. In the **bin** directory of the Flink client, run the ***sh generate_keystore.sh*** *<password>* command. For details, see **Configuring Flink Authentication and Encryption**. The configuration items in **Table 6-10** are set by default for MRS 3.*x* or later. You can also configure them manually.

Table 6-10 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**.

For versions earlier than MRS 3.x, the ***generate_keystore.sh*** command is generated automatically, and the configuration items in **Table 6-11** are set by default. You can also configure them manually.

Table 6-11 Parameter description

Parameter	Example Value	Description
security.ssl.internal.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.internal.keystore-password	-	A user-defined password of **keystore**.
security.ssl.internal.key-password	-	A user-defined password of the SSL key.
security.ssl.internal.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.internal.truststore-password	-	A user-defined password of **truststore**.

For versions earlier than MRS 3.x, if SSL for external connections is enabled, that is, **security.ssl.rest.enabled** is set to **true**, you need to configure the parameters listed in **Table 6-12**.

Table 6-12 Parameters

Parameter	Example Value	Description
security.ssl.rest.enabled	true	Enable external SSL. If this parameter is set to **true**, set the related parameters by referring to **Table 6-12**.
security.ssl.rest.keystore	${path}/flink.keystore	Path for storing the **keystore**.
security.ssl.rest.keystore-password	-	A user-defined password of **keystore**.
security.ssl.rest.key-password	-	A user-defined password of the SSL key.
security.ssl.rest.truststore	${path}/flink.truststore	Path for storing the **truststore**.
security.ssl.rest.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.

Using the Flink Client (MRS 3.*x* or Later)

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 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.
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.

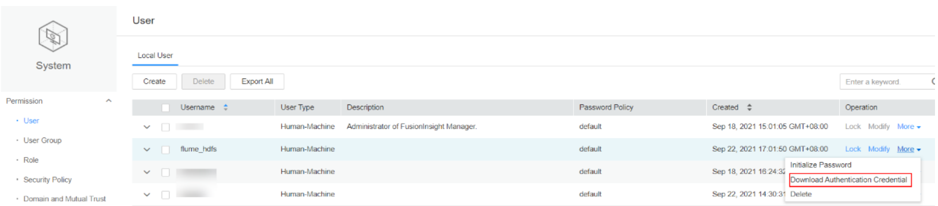
   Log in to FusionInsight Manager and choose **System** > **Permission** > **Role**. Click **Create Role** and configure **Role Name** and **Description**. In **Configure Resource Permission**, choose *Name of the desired cluster* > **Flink** and select **FlinkServer Admin Privilege**. Then click **OK**.

   Choose **System** > **Permission** > **User** and click **Create User**. Configure **Username**, set **User Type** to **Human-Machine**, configure **Password** and **Confirm Passowrd**, 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 FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **System** > **Permission** > **User**. In the **Operation** column of the created user, click **More** and select **Download Authentication Credential**.

   Figure 6-2 Downloading the authentication credential

3. Decompress the downloaded authentication credential package and copy the obtained file to a directory on the client node, for example, **/opt/hadoopclient/Flink/flink/conf**. If the client is installed on a node outside the cluster, copy the obtained file to the **/etc/** directory on this node.

4. Add the service IP address of the node where the client is installed and IP addresses of all master nodes 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 tree of the MRS management console, choose **Clusters > 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 command to perform security hardening and configure a password used 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. 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.

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

When a user submits or runs a job in Flink, the user must have the following permissions based on whether Ranger authentication is enabled for related services (such as HDFS and Kafka):

- 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 6-3 Job submitted successfully on Yarn

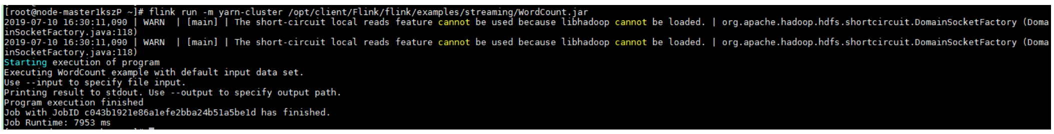


Figure 6-4 Session started successfully

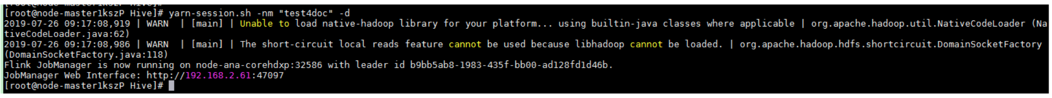
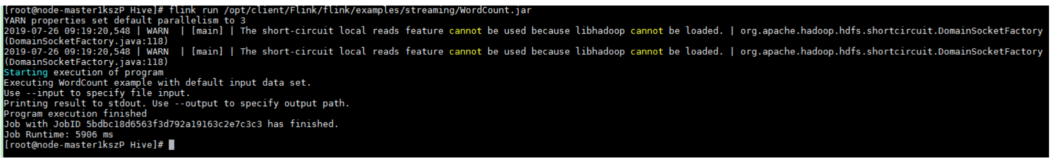


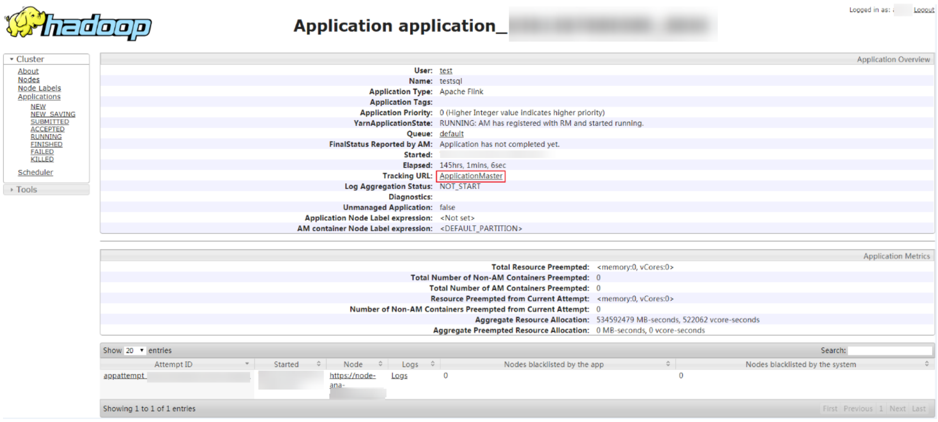
Figure 6-5 Job submitted successfully in the session



Step 8 Go to the native Yarn service page, find the application of the 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 6-6 application



----End

Using the Flink Client (Versions Earlier Than MRS 3.x)

Step 1 Install a client.

The following uses installing a Flink client on a core node as an example:

1. Log in to MRS Manager and choose **Services** > **Download Client** to download the client installation package to the active management node.
2. Use the IP address to search for the active management node, and log in to the active management node over VNC.
3. Log in to the active management node, and run the following command to switch the user:

   sudo su - omm
4. On the MRS management console, view the IP address on the **Nodes** tab page of the specified cluster.

   Record the IP address of the core node where the client is to be used.
5. On the active management node, run the following command to copy the client installation package to the core node:

   scp -p /tmp/MRS-client/MRS_Services_Client.tar *IP address of the core node*:**/opt/client**
6. Log in to the core node as user **root**.

   Master nodes support Cloud-Init. The preset username for Cloud-Init is **root**, and the password is the one set during cluster creation.
7. Run the following commands to install the client:

   cd /opt/client

   tar -xvf MRS_Services_Client.tar

tar -xvf MRS_Services_ClientConfig.tar

cd /opt/client/MRS_Services_ClientConfig

./install.sh *Client installation directory*

For example, run the following command:

./install.sh /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 If Kerberos authentication is enabled for the cluster, perform the following steps. If not, skip this whole step.

1. Prepare a user for submitting Flink jobs.

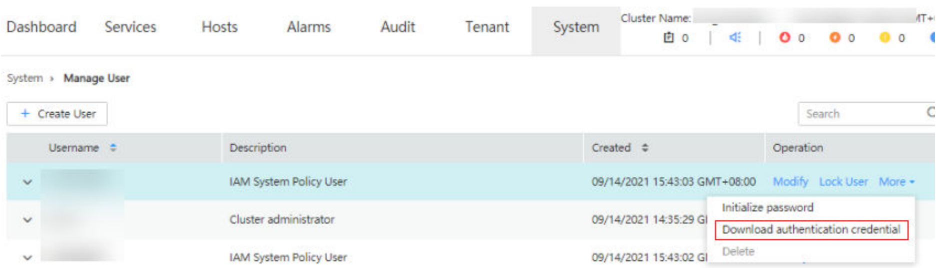
   Log in to MRS Manager and choose **System** > **Manage Role**. Click **Add Role** to add a role, for example, **flinkrole**. In **Permission**, choose **HDFS** > **File System** > **hdfs://hacluster/** and select **Read**, **Write**, and **Execute**. After you finish configuring this service, click **Service** in the **Permission** area. Choose **Yarn** > **Scheduler Queue** > **root**, select **Submit** for **default**, and click **OK**.

   Choose **System** > **Manage User Group**. Click **Create User Group** to create a user group for the sample project, for example, **flinkgroup**. Choose **System** > **Manage User**. Click **Create User** to create a user for the sample project. Enter a username, for example, **flinkuser**. Set **User Type** to **Human-machine**, add the user to user groups **flinkgroup** and **hadoop**, bind the role **flinkrole** to the user to obtain permissions, and click **OK**. (If you create a user for the first time, log in to MRS Manager as this user to change the password.)

2. Log in to Manager and download the authentication credential.

   Log in to Manager of the cluster. For details, see **Accessing MRS Manager (Versions Earlier Than MRS 3.x)**. Choose **System Settings** > **User Management**. In the **Operation** column of the row that contains the added user, choose **More** > **Download Authentication Credential**.

   Figure 6-7 Downloading the authentication credential



3. Decompress the downloaded authentication credential package and copy the obtained file to a directory on the client node, for example, **/opt/hadoopclient/Flink/flink/conf**. If the client is installed on a node outside the cluster, copy the obtained file to the **/etc/** directory on this node.

4. 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
```

5. Generate the **generate_keystore.sh** script by referring to **Example of Issuing a Certificate** and place it in the **bin** directory on the Flink client. In the **bin** directory of the Flink client, run the following command to perform security hardening and configure a password used for submitting jobs:

   cd /opt/hadoopclient/Flink/flink/bin

   sh generate_keystore.sh

   This script automatically replaces the SSL value in the **/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml** file. For MRS 2.*x* or earlier, external SSL is disabled for security clusters by default. To enable external SSL, configure it by referring to "Authentication and Encryption" and run this script again.

   NOTE

   – You do not need to manually generate the **generate_keystore.sh** script.
   – After authentication and encryption, the generated **flink.keystore**, **flink.truststore**, and **security.cookie** items are automatically filled in the corresponding configuration items in **flink-conf.yaml**.

6. 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.internal.keystore: ssl/flink.keystore
        security.ssl.internal.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

When a user submits or runs a job in Flink, the user must have the following permissions based on whether Ranger authentication is enabled for related services (such as HDFS and Kafka):

- 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 6-8 Job submitted successfully on Yarn

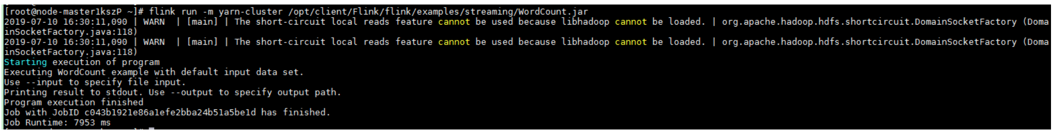


Figure 6-9 Session started successfully

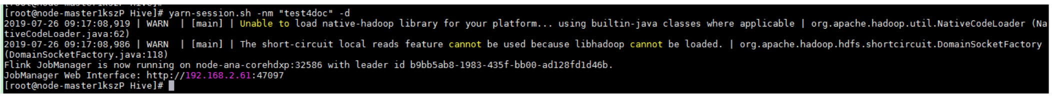
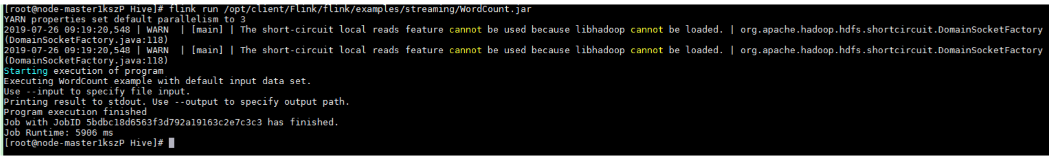


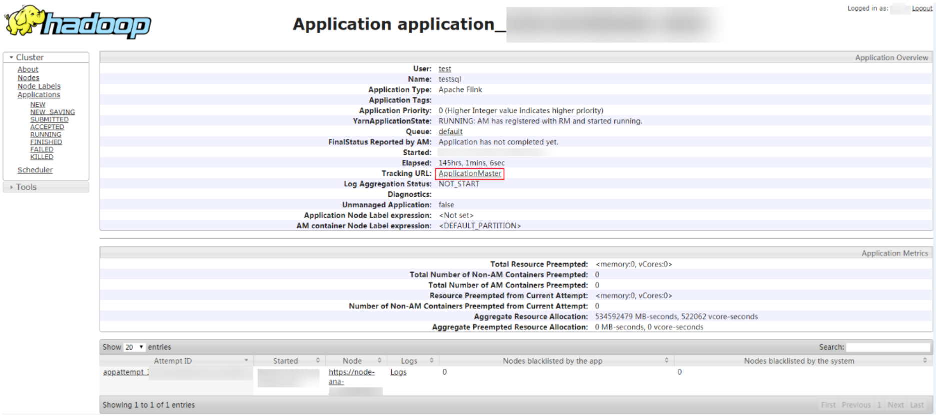
Figure 6-10 Job submitted successfully in the session



Step 8 Go to the native Yarn service page, find the application of the 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 6-11 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.

Impact on the System

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.

Procedure

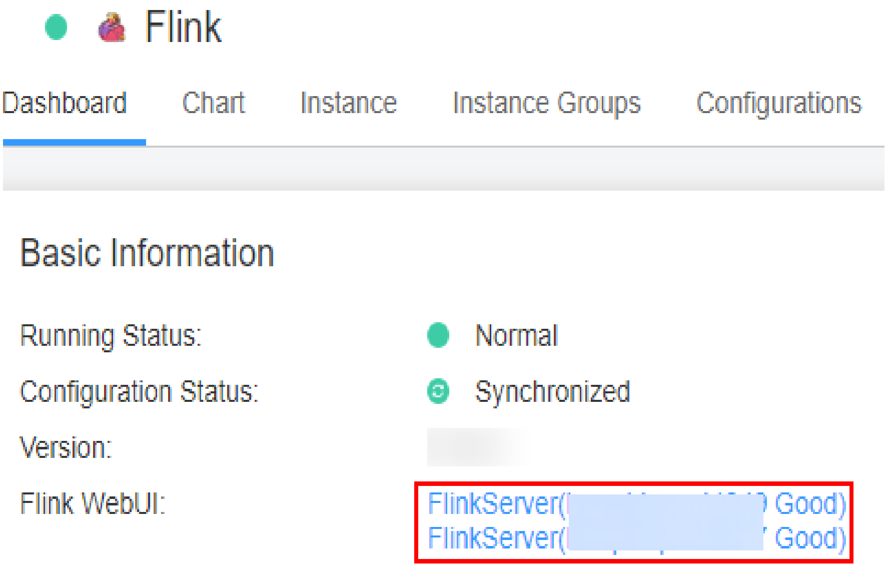
Step 1 Log in to FusionInsight Manager as a user with **FlinkServer Admin Privilege**. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > **Services** > **Flink**.

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 WebUI**, click the link to access the Flink web UI.

Figure 6-12 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

Scenario

Applications can be used to isolate different upper-layer services.

Creating an Application

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

Scenario

Different clusters can be accessed by configuring the cluster connection.

Creating a Cluster Connection

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 6-13** and click **OK**. After the cluster connection is created, you can edit, test, or delete the cluster connection in the **Operation** column.

Table 6-13 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

Scenario

You can use data connections to access different data services. Currently, FlinkServer supports HDFS and Kafka data connections.

Creating a Data Connection

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 6-14**, and click **OK**. After the data connection is created, you can edit, test, or delete the data connection in the **Operation** column.

Table 6-14 Parameters for creating a data connection

Parameter	Description	Example Value
Data Connection Type	Type of the data connection, which can be **HDFS** or **Kafka**.	-
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
Authentication Mode	● **SIMPLE**: indicates that the connected service is in non-security mode and does not need to be authenticated. ● **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

Scenario

Data tables can be used to define basic attributes and parameters of source tables, dimension tables, and output tables.

Creating a Stream Table

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 6-15** and click **OK**. After the stream table is created, you can edit or delete the stream table in the **Operation** column.

Table 6-15 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**	-
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**	-

Parameter	Description	Remarks
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**.	-
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

Scenario

Define Flink jobs, including Flink SQL and Flink JAR jobs.

Creating a Job

Step 1 Access the Flink web UI. For details, see **Accessing the FlinkServer Web UI**.

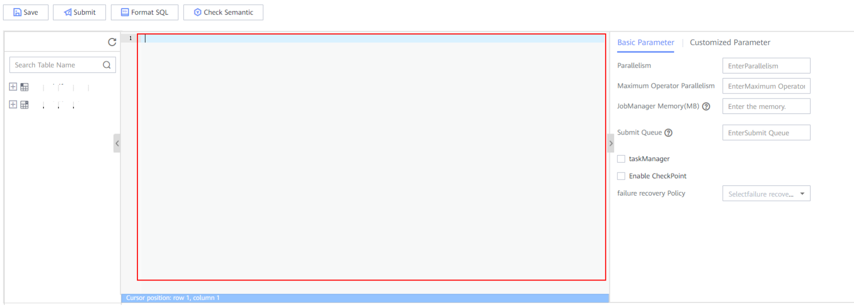
Step 2 Click **Job Management**. The job management page is displayed.

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.

- Creating a Flink SQL job

  a. Develop the job on the job development page.



  b. Click **Check Semantic** to check the input content and click **Format SQL** to format SQL statements.

  c. After the job SQL statements are developed, set basic and customized parameters as required by referring to **Table 6-16** and click **Save**.

Table 6-16 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 **512**.
Submit Queue	Queue to which a job is submitted. If this parameter is not set, the **default** queue is used.
taskManager	taskManager running parameters include: ▪ **Slots**: If this parameter is left blank, the default value **1** is used. ▪ **Memory (MB)**: The minimum value is **512**.

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. **EXACTLY_ONCE**: Data or events are processed by the operator only once. **AT_LEAST_ONCE**: Data or events are processed by the operator 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 6-17** or add customized parameters.

Table 6-17 Parameter configuration

Parameter	Description
Local .jar File	Upload a local JAR file. The size of the file cannot exceed 10 MB.

Parameter	Description
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.
Type	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
JobManager Memory (MB)	Memory of JobManager The minimum value is **512**.
Submit Queue	Queue to which a job is submitted. If this parameter is not set, the **default** queue is used.
taskManager	taskManager running parameters include: ■ **Slots**: If this parameter is left blank, the default value **1** is used. ■ **Memory (MB)**: The minimum value is **512**.

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.

----End

6.6 Managing FlinkServer Jobs

6.6.1 Configuring the FlinkServer Job Restart Policy

Overview

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
```

Choosing 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
  ```

6.6.2 Using UDFs in FlinkServer Jobs

This section applies to MRS 3.1.2 or later.

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 6-18**.

Table 6-18 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**.

Prerequisites

You have prepared a UDF JAR file whose size does not exceed 200 MB.

Uploading a UDF

Step 1 Access the Flink web UI. For details, see **Accessing the FlinkServer Web UI**.

Step 2 Click **UDF Management**. The **UDF Management** page is displayed.

Step 3 Click **Add UDF**. Select and upload the prepared UDF JAR file for **Local .jar File**.

Step 4 Enter the UDF name and description and click **OK**.

NOTE

- A maximum of 10 UDF names can be added. **Name** can be customized. **Type** must correspond to the UDF fully qualified class name 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 5 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 6 (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);
```

6.7 Flink O&M Management

6.7.1 Typical Flink Configuration Parameters

Description

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*.

Configurations

The following describes the parameters:

- **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 timeout settings for message sending and waiting and the Akka DeathWatch detection meachanism.
- **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 monitoring port, whether to enable SSL encryption, domain of the network used for NettySink monitoring.

JobManager & TaskManager

Table 6-19 JobManager & TaskManager parameters

Parameter	Description	Default Value	Mandatory	Remarks
taskmanager.memory.size	Amount of heap memory of the Java virtual machine (JVM) that TaskManager reserves for sorting, hash tables, and caching of intermediate results. If unspecified, the memory manager will take a fixed ratio with respect to the size of JVM as specified by **taskmanager.memory.fraction**. The unit is MB.	0	No	Only versions earlier than MRS 3.*x*
taskmanager.registration.initial-backoff	Initial interval between two consecutive registration attempts. The unit is ms/s/m/h/d. 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.registration.refused-backoff	Retry interval when a registration connection is rejected by JobManager.	5 min	No	

Parameter	Description	Default Value	Mandatory	Remarks
taskmanager.rpc.port	IPC port range of TaskManager	32326-32390	No	Applicable to all versions
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	
taskmanager.memory.fraction	Ratio of JVM heap memory that TaskManager reserves for sorting, hash tables, and caching of intermediate status.	0.7	No	
taskmanager.memory.off-heap	Whether TaskManager uses off-heap memory for sorting, hash tables, and caching of intermediate status. You are advised to enable this item for large memory needs to improve memory operation efficiency.	false	Yes	
taskmanager.memory.preallocate	Whether TaskManager allocates reserved memory upon startup. You are advised to enable this item when off-heap memory is used.	false	No	

Parameter	Description	Default Value	Mandatory	Remarks
task.cancellation.interval	Interval between two successive task cancellation attempts. The unit is millisecond.	30000	No	
client.rpc.port	Akka system monitoring port on the Flink client.	32651-32720	No	Only MRS 3.*x* or later
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	

Parameter	Description	Default Value	Mandatory	Remarks
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	
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 an idle slot in Slot Pool, in milliseconds.	50000	No	
slot.request.timeout	Timeout for requesting a slot from Slot Pool, in milliseconds.	300000	No	

Parameter	Description	Default Value	Mandatory	Remarks
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	
taskmanager.network.memory.buffers-per-channel	Maximum number of network buffers used by each output/input channel (sub-partition/incoming channel). In credit-based flow control mode, this indicates how much credit is in each input channel. 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	

Parameter	Description	Default Value	Mandatory	Remarks
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	
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 (in milliseconds) for partition requests of input channels.	100	No	
taskmanager.network.request-backoff.max	Maximum backoff (in milliseconds) 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 6-20 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
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. This parameter is available only to MRS 3.*x* or later.	3600	No

Distributed Coordination (via Akka)

Table 6-21 Distributed Coordination parameters

Parameter	Description	Default Value	Mandatory	Remarks
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	Applicable to all versions
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	
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	

Parameter	Description	Default Value	Mandatory	Remarks
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	Applicable only to MRS 3.*x* or later
akka.client-socket-worker-pool.pool-size-max	Maximum number of threads calculated based on the factor.	2	No	

Parameter	Description	Default Value	Mandatory	Remarks
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 6-22 SSL parameters

Parameter	Description	Default Value	Mandatory	Remarks
security.ssl.internal.enabled	Whether to enable SSL for internal communication. This parameter is automatically configured based on the security mode of the cluster.	• Security mode: **true** • Normal mode: **false**	Yes	Applicable only to versions earlier than MRS 3.*x*
security.ssl.internal.keystore	Java keystore file.	-	Yes	
security.ssl.internal.keystore-password	Password used to decrypt the keystore file.	-	Yes	
security.ssl.internal.key-password	Password used to decrypt the server key in the keystore file.	-	Yes	
security.ssl.internal.truststore	**truststore** file containing the public CA certificates.	-	Yes	
security.ssl.internal.truststore-password	Password used to decrypt the truststore file.	-	Yes	
security.ssl.rest.enabled	Whether to enable SSL for external communication. This parameter is automatically configured based on the security mode of the cluster.	• Security mode: **true** • Normal mode: **false**	Yes	
security.ssl.rest.keystore	Java keystore file.	-	Yes	
security.ssl.rest.keystore-password	Password used to decrypt the keystore file.	-	Yes	
security.ssl.rest.key-password	Password used to decrypt the server key in the keystore file.	-	Yes	
security.ssl.rest.truststore	**truststore** file containing the public CA certificates.	-	Yes	
security.ssl.rest.truststore-password	Password used to decrypt the truststore file.	-	Yes	

Parameter	Description	Default Value	Mandatory	Remarks
security.ssl.protocol	SSL transmission protocol version.	TLSv1.2	Yes	Applicable to all versions
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	Applicable only to MRS 3.*x* or later
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 6-23 Network communication parameters

Parameter	Description	Default Value	Mandatory
taskmanager.network.netty.num-arenas	Number of Netty memory blocks.	1	No
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 6-24 JobManager Web Frontend parameters

Parameter	Description	Default Value	Mandatory	Remarks
jobmanager.web.port	The web port. Value range: 32261–32325.	32261-32325	No	Applicable only to versions earlier than MRS 3.*x*
jobmanager.web.allow-access-address	Web access whitelist. IP addresses are separated by commas (,). Only whitelisted IP addresses can access the web.	*	Yes	Applicable to all versions
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	Applicable only to MRS 3.*x* or later
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	

Parameter	Description	Default Value	Mandatory	Remarks
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	
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**: No content is saved to the cache.	Yes	

Parameter	Description	Default Value	Mandatory	Remarks
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. * Indicates that any website can access the service port across domains. You can set this parameter to a specified website.	* (non-security cluster)	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 page switch to a specified page.	Automatic configuration	Yes	
jobmanager.web.404-redirect-url	Web page access error 404. If 404 error occurs, the page switch to a specified page.	Automatic configuration	Yes	
jobmanager.web.415-redirect-url	Web page access error 415. If 415 error occurs, the page switch to a specified page.	Automatic configuration	Yes	
jobmanager.web.500-redirect-url	Web page access error 500. If 500 error occurs, the page switch to a specified page.	Automatic configuration	Yes	
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	

Parameter	Description	Default Value	Mandatory	Remarks
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 6-25 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
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 6-26 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
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

Parameter	Description	Default Value	Mandatory
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
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 6-27 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
security.kerberos.login.contexts	Contexts of the jass file generated by Flink. This parameter is a server parameter.	Client, KafkaClient	Yes

HA

Table 6-28 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

Parameter	Description	Default Value	Mandatory
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
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. This parameter is available only to MRS 3.*x* or later.	The default value is the same as the value of **akka.ask.timeout**.	No

Parameter	Description	Default Value	Mandatory
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. For ACL options, see **https://zookeeper.apache.org/doc/r3.5.1-alpha/zookeeperProgrammers.html#sc_BuiltinACLSchemes**.	• 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 6-29 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 6-30 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

It applies to MRS 3.*x* or later.

Table 6-31 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.

Parameter	Description	Default Value	Mandatory
nettyconnector.sinkserver.port.range	Port range of NettySink.	28444-28843	No. However, if pipeline is enabled, the feature is 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.

6.7.2 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**

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.

NOTE

For versions earlier than MRS 3.x, The executor logs are backed up each time when the log size reaches 30 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 6-32 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 and audit logs.
   - By default, FlinkServer service 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 6-33 FlinkServer log list

<table>
<tr><th>Type</th><th>Name</th><th>Description</th></tr>
<tr><td rowspan="5">FlinkServer run logs</td><td>checkService.log</td><td>Health check log</td></tr>
<tr><td>checkFlinkServer.log</td><td>Health check log of FlinkServer</td></tr>
<tr><td>localhost_access_log..yyyy-mm-dd.txt</td><td>URL log of FlinkServer</td></tr>
<tr><td>start_thrift_server.out</td><td rowspan="2">Thrift server startup log</td></tr>
<tr><td>thrift_server_thriftServer_xxx.log.last</td></tr>
</table>

Type	Name	Description
	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	
FlinkServer audit logs	flinkserver_audit_*yyyymmdd-x*.log.gz	Audit archive log
	flinkserver_audit.log	Audit log

Log Level

Table 6-34 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 6-34 Log levels

Level	Description
ERROR	Error information about the current event processing

Level	Description
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 6-35 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)

6.8 Flink Performance Tuning

6.8.1 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.

6.8.2 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.

6.8.3 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

6.8.4 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 Typical Commands of the Flink Client

This section applies to MRS 3.*x* or later.

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 Install the Flink client in **/opt/client**.

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 6-36**.

Table 6-36 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. **-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). **-yqu,--yarnqueue <arg>**: Specifies a Yarn queue.	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
	-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.
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).

Command	Description	Description
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

6.10 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.

6.11 Example of Issuing a Certificate

Generate the **generate_keystore.sh** script based on the sample code and save the script to the **bin** directory on the Flink client.

```
#!/bin/bash

KEYTOOL=${JAVA_HOME}/bin/keytool
KEYSTOREPATH="$FLINK_HOME/conf/"
CA_ALIAS="ca"
CA_KEYSTORE_NAME="ca.keystore"
CA_DNAME="CN=Flink_CA"
CA_KEYALG="RSA"
CLIENT_CONF_YAML="$FLINK_HOME/conf/flink-conf.yaml"
KEYTABPRINCEPAL=""

function getConf()
{
    if [ $# -ne 2 ]; then
        echo "invalid parameters for getConf"
        exit 1
```

```
    fi

    confName="$1"
    if [ -z "$confName" ]; then
        echo "conf name is empty."
        exit 2
    fi

    configFile=$FLINK_HOME/conf/client.properties
    if [ ! -f $configFile ]; then
        echo $configFile" is not exist."
        exit 3
    fi

    defaultValue="$2"
    cnt=$(grep $1 $configFile | wc -l)
    if [ $cnt -gt 1 ]; then
        echo $confName" has multi values in "$configFile
        exit 4
    elif [ $cnt -lt 1 ]; then
        echo $defaultValue
    else
        line=$(grep $1 $configFile)
        confValue=$(echo "${line#*=}")
        echo "$confValue"
    fi
}

function createSelfSignedCA()
{
    #varible from user input
    keystorePath=$1
    storepassValue=$2
    keypassValue=$3

    #generate ca keystore
    rm -rf $keystorePath/$CA_KEYSTORE_NAME
    $KEYTOOL -genkeypair -alias $CA_ALIAS -keystore $keystorePath/$CA_KEYSTORE_NAME -dname 
$CA_DNAME -storepass $storepassValue -keypass $keypassValue -validity 3650 -keyalg $CA_KEYALG -
keysize 3072 -ext bc=ca:true
    if [ $? -ne 0 ]; then
        echo "generate ca.keystore failed."
        exit 1
    fi

    #generate ca.cer
    rm -rf "$keystorePath/ca.cer"
    $KEYTOOL -keystore "$keystorePath/$CA_KEYSTORE_NAME" -storepass "$storepassValue" -alias 
$CA_ALIAS -validity 3650 -exportcert > "$keystorePath/ca.cer"
    if [ $? -ne 0 ]; then
        echo "generate ca.cer failed."
        exit 1
    fi

    #generate ca.truststore
    rm -rf "$keystorePath/flink.truststore"
    $KEYTOOL -importcert -keystore "$keystorePath/flink.truststore" -alias $CA_ALIAS -storepass 
"$storepassValue" -noprompt -file "$keystorePath/ca.cer"
    if [ $? -ne 0 ]; then
        echo "generate ca.truststore failed."
        exit 1
    fi
}

function generateKeystore()
{
    #get path/pass from input
    keystorePath=$1
    storepassValue=$2
```

```
    keypassValue=$3

    #get value from conf
    aliasValue=$(getConf "flink.keystore.rsa.alias" "flink")
    validityValue=$(getConf "flink.keystore.rsa.validity" "3650")
    keyalgValue=$(getConf "flink.keystore.rsa.keyalg" "RSA")
    dnameValue=$(getConf "flink.keystore.rsa.dname" "CN=flink.com")
    SANValue=$(getConf "flink.keystore.rsa.ext" "ip:127.0.0.1")
    SANValue=$(echo "$SANValue" | xargs)
    SANValue="ip:$(echo "$SANValue"| sed 's/,/,ip:/g')"

    #generate keystore
    rm -rf $keystorePath/flink.keystore
    $KEYTOOL -genkeypair -alias $aliasValue -keystore $keystorePath/flink.keystore -dname $dnameValue -
ext SAN=$SANValue -storepass $storepassValue -keypass $keypassValue -keyalg $keyalgValue -keysize
3072 -validity 3650
    if [ $? -ne 0 ]; then
        echo "generate flink.keystore failed."
        exit 1
    fi

    #generate cer
    rm -rf $keystorePath/flink.csr
    $KEYTOOL -certreq -keystore $keystorePath/flink.keystore -storepass $storepassValue -alias $aliasValue -
file $keystorePath/flink.csr
    if [ $? -ne 0 ]; then
        echo "generate flink.csr failed."
        exit 1
    fi

    #generate flink.cer
    rm -rf $keystorePath/flink.cer
    $KEYTOOL -gencert -keystore $keystorePath/ca.keystore -storepass $storepassValue -alias $CA_ALIAS -
ext SAN=$SANValue -infile $keystorePath/flink.csr -outfile $keystorePath/flink.cer -validity 3650
    if [ $? -ne 0 ]; then
        echo "generate flink.cer failed."
        exit 1
    fi

    #import cer into keystore
    $KEYTOOL -importcert -keystore $keystorePath/flink.keystore -storepass $storepassValue -file
$keystorePath/ca.cer -alias $CA_ALIAS -noprompt
    if [ $? -ne 0 ]; then
        echo "importcert ca."
        exit 1
    fi

    $KEYTOOL -importcert -keystore $keystorePath/flink.keystore -storepass $storepassValue -file
$keystorePath/flink.cer -alias $aliasValue -noprompt;
    if [ $? -ne 0 ]; then
        echo "generate flink.truststore failed."
        exit 1
    fi
}

function configureFlinkConf()
{
    # set config
    if [ -f "$CLIENT_CONF_YAML" ]; then
        SSL_ENCRYPT_ENABLED=$(grep "security.ssl.encrypt.enabled" "$CLIENT_CONF_YAML" | awk '{print
$2}')
        if [ "$SSL_ENCRYPT_ENABLED" = "false" ];then

            sed -i s/"security.ssl.key-password:".*/"security.ssl.key-password:"\ "${keyPass}"/g
"$CLIENT_CONF_YAML"
            if [ $? -ne 0 ]; then
                echo "set security.ssl.key-password failed."
                return 1
            fi
```

```
            sed -i s/"security.ssl.keystore-password:".*/"security.ssl.keystore-password:"\ "${storePass}"/g
"$CLIENT_CONF_YAML"
            if [ $? -ne 0 ]; then
                echo "set security.ssl.keystore-password failed."
                return 1
            fi

            sed -i s/"security.ssl.truststore-password:".*/"security.ssl.truststore-password:"\ "${storePass}"/g
"$CLIENT_CONF_YAML"
            if [ $? -ne 0 ]; then
                echo "set security.ssl.keystore-password failed."
                return 1
            fi

            echo "security.ssl.encrypt.enabled is false, set security.ssl.key-password security.ssl.keystore-
password security.ssl.truststore-password success."
        else
            echo "security.ssl.encrypt.enabled is true, please enter security.ssl.key-password security.ssl.keystore-
password security.ssl.truststore-password encrypted value in flink-conf.yaml."
        fi

        keystoreFilePath="${keystorePath}"/flink.keystore
        sed -i 's#'"security.ssl.keystore:".*'#'"security.ssl.keystore:"\ "$keystoreFilePath"'#g'
"$CLIENT_CONF_YAML"
        if [ $? -ne 0 ]; then
            echo "set security.ssl.keystore failed."
            return 1
        fi


        truststoreFilePath="${keystorePath}/flink.truststore"
        sed -i 's#'"security.ssl.truststore:".*'#'"security.ssl.truststore:"\ "$truststoreFilePath"'#g'
"$CLIENT_CONF_YAML"
        if [ $? -ne 0 ]; then
            echo "set security.ssl.truststore failed."
            return 1
        fi

        command -v sha256sum >/dev/null
        if [ $? -ne 0 ];then
            echo "sha256sum is not exist, it will produce security.cookie with date +%F-%H-%M-%s-%N."
            cookie=$(date +%F-%H-%M-%s-%N)
        else
            cookie="$(echo "${KEYTABPRINCEPAL}"| sha256sum | awk '{print $1}')"
        fi

        sed -i s/"security.cookie:".*/"security.cookie:"\ "${cookie}"/g "$CLIENT_CONF_YAML"
        if [ $? -ne 0 ]; then
            echo "set security.cookie failed."
            return 1
        fi
    fi
    return 0;
}

main()
{
    #check environment variable is set or not
    if [ -z ${FLINK_HOME+x} ]; then
        echo "errro: environment variables are not set."
        exit 1
    fi
    stty -echo
    read -rp "Enter password:" password
    stty echo
    echo
```

```
    KEYTABPRINCEPAL=$(grep "security.kerberos.login.principal" "$CLIENT_CONF_YAML" | awk '{print $2}')
    if [ -z "$KEYTABPRINCEPAL" ];then
        echo "please config security.kerberos.login.principal info first."
        exit 1
    fi


    #get input
    keystorePath="$KEYSTOREPATH"
    storePass="$password"
    keyPass="$password"

    #generate self signed CA
    createSelfSignedCA "$keystorePath" "$storePass" "$keyPass"
    if [ $? -ne 0 ]; then
        echo "create self signed ca failed."
        exit 1
    fi

    #generate keystore
    generateKeystore "$keystorePath" "$storePass" "$keyPass"
    if [ $? -ne 0 ]; then
        echo "create keystore failed."
        exit 1
    fi

    echo "generate keystore/truststore success."

    # set flink config
    configureFlinkConf "$keystorePath" "$storePass" "$keyPass"
    if [ $? -ne 0 ]; then
        echo "configure Flink failed."
        exit 1
    fi

    return 0;
}

#the start main
main "$@"

exit 0
```


NOTE

Run the **sh generate_keystore.sh** *<password>* command. *<password>* is user-defined.

- If *<password>* contains the special character **$**, use the following method to avoid the password being escaped: **sh generate_keystore.sh 'Bigdata_2013'**. 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.
- The password cannot contain **#**.
- Before using the **generate_keystore.sh** script, run the **source bigdata_env** command in the client directory.
- When the **generate_keystore.sh** script is used, the absolute paths of **security.ssl.keystore** and **security.ssl.truststore** are automatically filled in **flink-conf.yaml**. Therefore, you need to manually change the paths to relative paths as required. Example:
  - Change **/opt/client/Flink/flink/conf//flink.keystore** to **security.ssl.keystore: ssl/flink.keystore**.
  - Change **/opt/client/Flink/flink/conf//flink.truststore** to **security.ssl.truststore: ssl/flink.truststore**.
  - Create the **ssl** folder in any directory on the Flink client. For example, create the **ssl** folder in the **/opt/client/Flink/flink/conf/** directory and save the **flink.keystore** and **flink.truststore** files to the **ssl** folder.
  - When you run the **yarn-session** or **flink run -m yarn-cluster** command, run the **yarn-session.sh -t ssl -d** or **flink run -m yarn-cluster -yt ssl -d WordCount.jar** command in the same directory as the **ssl** folder.

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.

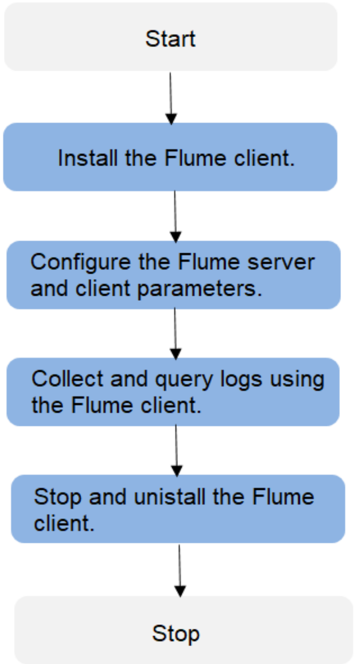
The client-side FlumeAgent needs to be independently installed. Data can be directly imported to components such as HDFS and Kafka. Additionally, the client-side and service-side FlumeAgents can also work together to provide services.

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

This section applies to MRS 3.*x* or later.

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**.
Selector.header	Flume.selector.header	-
Selector.default	-	-
Selector.mapping.*	-	-

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.

SinkGroup

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.
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 custom mechanism 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.

Parameter	Default Value	Description
processor.type	default	Specifies the type of a processor. Set this parameter to **failover**.
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

1. 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.
2. Flume allows multiple interceptors to be configured for a source. The interceptor names are separated by spaces.
3. The specified interceptor sequence is the order in which they are called.
4. 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.

Interceptor Type	Description
Remove Header Interceptor	The interceptor discards the corresponding event based on the strings that matches the regular expression contained in the event header.
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
```

7.3 Installing the Flume Client

7.3.1 Installing the Flume Client on Clusters of Versions Earlier Than MRS 3.x

Scenario

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.

This section applies to MRS 3.*x* or earlier clusters.

Prerequisites

- A streaming 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.

Procedure

Step 1 Create an ECS that meets the requirements.

Step 2 Go to the cluster details page and choose **Components**.

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 3 Click **Download Client**.

1. In **Client Type**, select **All client files**.
2. In **Download to**, select **Remote host**.
3. Set **Host IP Address** to the IP address of the ECS, **Host Port** to **22**, and **Save Path** to **/tmp**.
   - If the default port **22** for logging in to an ECS through SSH has been changed, set **Host Port** to a new port.
   - The value of **Save Path** contains a maximum of 256 characters.
4. Set **Login User** to **root**.

   If another user is used, ensure that the user has permissions to read, write, and execute the save path.
5. Select **Password** or **SSH Private Key** for **Login Mode**.
   - **Password**: Enter the password of user **root** set during cluster creation.
   - **SSH Private Key**: Select and upload the key file used for creating the cluster.
6. Click **OK** to generate a client file.

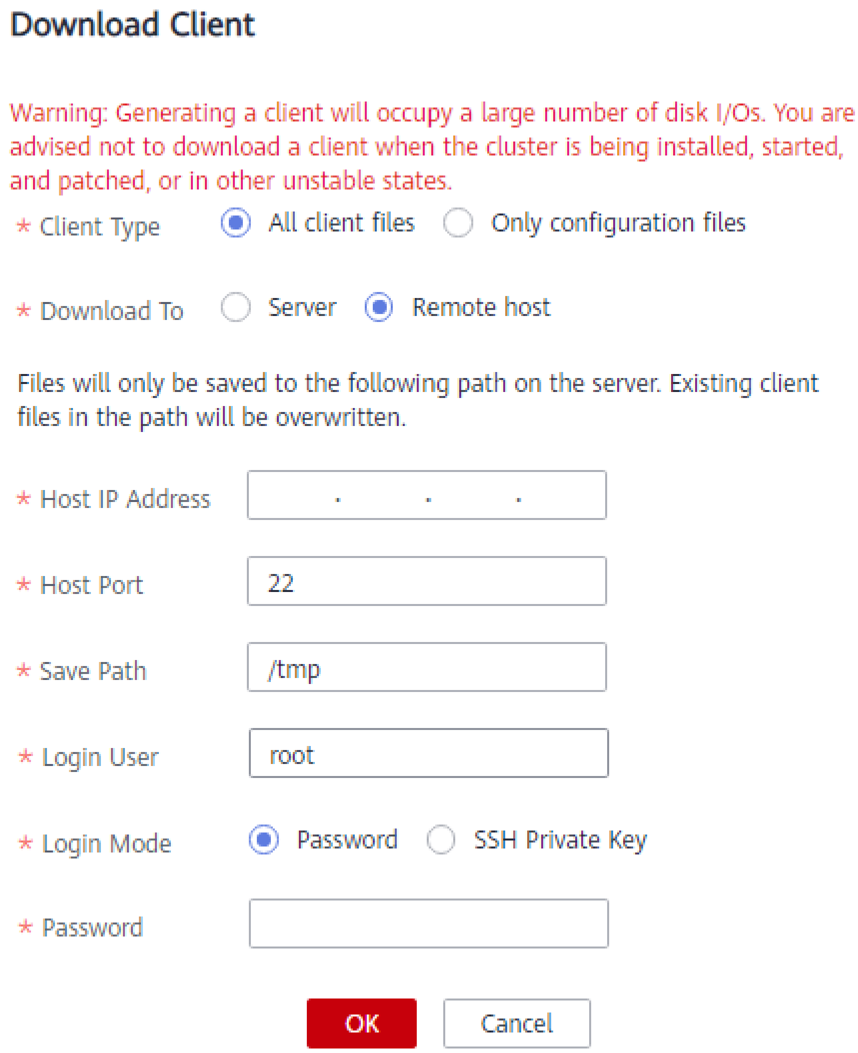
   If the following information is displayed, the client package is saved.

   ```
   Client files downloaded to the remote host successfully.
   ```

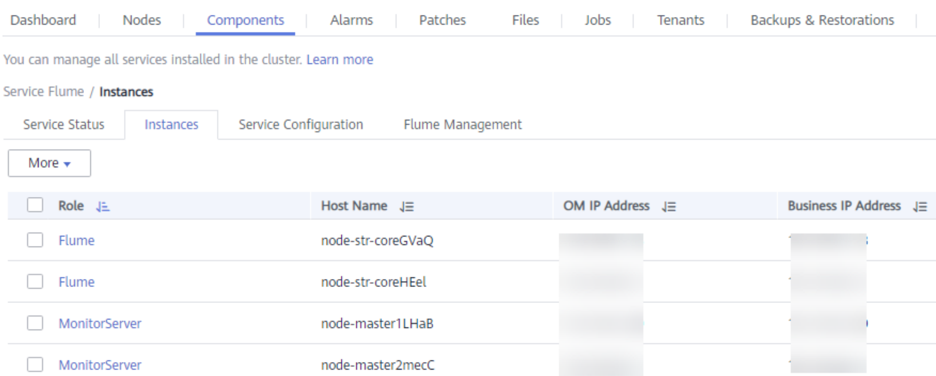
   If the following information is displayed, check the username, password, and security group configurations of the remote host. Ensure that the username and password are correct and an inbound rule of the SSH (22) port has been added to the security group of the remote host. And then, go to **Step 3** to download the client again.

   ```
   Failed to connect to the server. Please check the network connection or parameter settings.
   ```

Figure 7-2 Downloading a client



Step 4 Choose **Flume** > **Instance**. Query the **Business IP Address** of any Flume instance and any two MonitorServer instances.



Step 5 Log in to the ECS using VNC. For details, see "**User Guide**" > "**Login Using VNC**" in *Elastic Cloud Server*. ("**Instances**" > "**Logging In to a Linux ECS**").

All images support Cloud-Init. The preset username for Cloud-Init is **root** and the password is the one you set during cluster creation. You are advised to change the password upon the first login.

Step 6 On the ECS, switch to user **root** and copy the installation package to the **/opt** directory.

sudo su - root

cp /tmp/MRS_Flume_Client.tar /opt

Step 7 Run the following command in the **/opt** directory to decompress the package and obtain the verification file and the configuration package of the client:

tar -xvf MRS_Flume_Client.tar

Step 8 Run the following command to verify the configuration package of the client:

sha256sum -c MRS_Flume_ClientConfig.tar.sha256

If the following information is displayed, the file package is successfully verified:

```
MRS_Flume_ClientConfig.tar: OK
```

Step 9 Run the following command to decompress **MRS_Flume_ClientConfig.tar**:

tar -xvf MRS_Flume_ClientConfig.tar

Step 10 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/MRS_Flume_ClientConfig/install.sh /opt/Flumeenv

If the following information is displayed, the client running environment is successfully installed:

```
Components client installation is complete.
```

Step 11 Run the following command to configure environment variables:

source /opt/Flumeenv/bigdata_env

Step 12 Run the following commands to decompress the Flume client package:

cd /opt/MRS_Flume_ClientConfig/Flume

tar -xvf FusionInsight-Flume-1.6.0.tar.gz

Step 13 Run the following command to check whether the password of the current user has expired:

chage -l root

If the value of **Password expires** is earlier than the current time, the password has expired. Run the **chage -M -1 root** command to validate the password.

Step 14 Run the following command to install the Flume client to a new directory, for example, **/opt/FlumeClient**. A directory is automatically generated during the client installation.

sh /opt/MRS_Flume_ClientConfig/Flume/install.sh -d /opt/FlumeClient -f *service IP address of the MonitorServer instance* **-c** *path of the Flume* *configuration file* **-l /var/log/ -e** *service IP address of Flume* **-n** *name of the Flume client*

The parameters are described as follows:

- **-d**: indicates the installation path of the Flume client.
- (Optional) **-f**: indicates the service IP addresses of the two MonitorServer instances, separated by a comma (,). If the IP addresses are not configured, the Flume client will not send alarm information to MonitorServer, and the client information will not be displayed on MRS Manager.
- (Optional) **-c**: indicates the **properties.properties** configuration file that the Flume client loads after installation. If this parameter is not specified, the **fusioninsight-flume-1.6.0/conf/properties.properties** file in the client installation directory is used by default. The configuration file of the client is empty. You can modify the configuration file as required and the Flume client will load it automatically.
- (Optional) **-l**: indicates the log directory. The default value is **/var/log/Bigdata**.
- (Optional) **-e**: indicates the service IP address of the Flume instance. It is used to receive the monitoring indicators reported by the client.
- (Optional) **-n**: indicates the name of the Flume client.
- IBM JDK does not support **-Xloggc**. You must change **-Xloggc** to **-Xverbosegclog** in **flume/conf/flume-env.sh**. For 32-bit JDK, the value of **-Xmx** must not exceed 3.25 GB.
- In **flume/conf/flume-env.sh**, the default value of **-Xmx** is 4 GB. If the client memory is too small, you can change it to 512 MB or even 1 GB.

For example, run **sh install.sh -d /opt/FlumeClient**.

If the following information is displayed, the client is successfully installed:

```
install flume client successfully.
```

----End

7.3.2 Installing the Flume Client on MRS 3.*x* or Later Clusters

Scenario

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.

This section applies to MRS 3.*x* or later.

Prerequisites

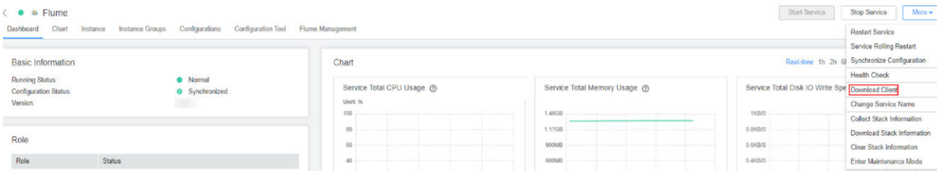
- A cluster with the Flume component has been created.
- The log host is in the same VPC and subnet with the MRS cluster.
- You have obtained the username and password for logging in to the log host.
- The installation directory is automatically created if it does not exist. If it exists, the directory must be left blank. The directory path cannot contain any space.

Procedure

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 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. 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
  ```
- IBM JDK does not support **-Xloggc**. You must change **-Xloggc** to **-Xverbosegclog** in **flume/conf/flume-env.sh**. For 32-bit JDK, the value of **-Xmx** must not exceed 3.25 GB.
- 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.

----End

Flume Client Cgroup Usage Guide

- **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

Scenario

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 (Versions Earlier Than MRS 3.x)

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 on Clusters of Versions Earlier Than MRS 3.x**. 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.

${BIGDATA_HOME}/MRS_Current/1_*X***_KerberosClient/etc/kdc.conf** is used as the full file path.

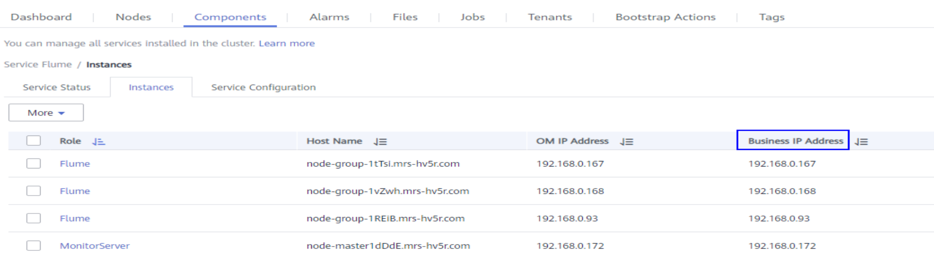
In the preceding paths, **X** indicates a random number. Change it based on the site requirements. The file must be saved by the user who installs the Flume client, for example, user **root**.

Step 3 Check the service IP address of any node where the Flume role is deployed.

Log in to the cluster details page, choose *Name of the desired cluster* > **Components** > **Flume** > **Instances**, and 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.

${BIGDATA_HOME}/MRS_*XXX***/install/FusionInsight-Flume-***Flume component version number*/**flume/conf/flume.keytab** is used as the full file path.

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.

${BIGDATA_HOME}/MRS_Current/1_*X*_Flume/etc/jaas.conf is used as the full file path.

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 Run the following command to configure and save jobs in the Flume client configuration file **properties.properties** based on service requirements.

vi *Flume client installation directory***/fusioninsight-flume-***Flume component version number***/conf/properties.properties**

The following uses SpoolDir Source+File Channel+Kafka Sink as an example:

```
##########################################################################
###############
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
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.

  Obtain the value by checking **${BIGDATA_HOME}/MRS_Current/1_*X*_Broker/etc/server.properties** on the node where the broker instance resides.

  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

Using the Flume Client (MRS 3.*x* or Later)

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 on MRS 3.x or Later Clusters**. 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**. 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

Step 9 Configure jobs based on actual service scenarios.

- Some parameters, for MRS 3.*x* or later, can be configured on Manager.
- 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.

  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
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.

  Obtain the value by checking **${BIGDATA_HOME}/MRS_Current/1_*X*_Broker/etc/server.properties** on the node where the broker instance resides.

  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

Scenario

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.

This section applies to MRS 3.*x* or later.

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 and Flume service have been installed.
- The network environment of the cluster is secure.

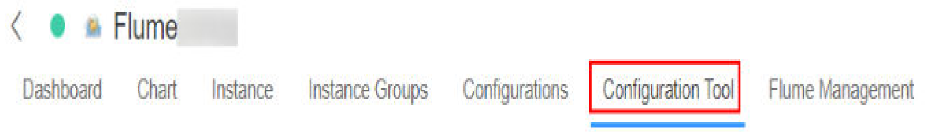
Procedure

Step 1 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**.

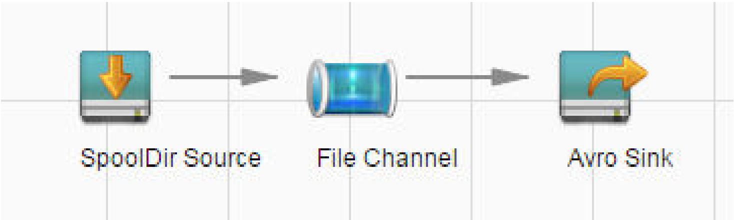
   Figure 7-3 Clicking Configuration Tool

   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 7-4**.

   Figure 7-4 Example for the Flume configuration tool

   c. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-8** 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 7-8 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 2 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 7-5 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 7-6**.

Figure 7-6 Example for the Flume configuration tool

c. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-9** 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.

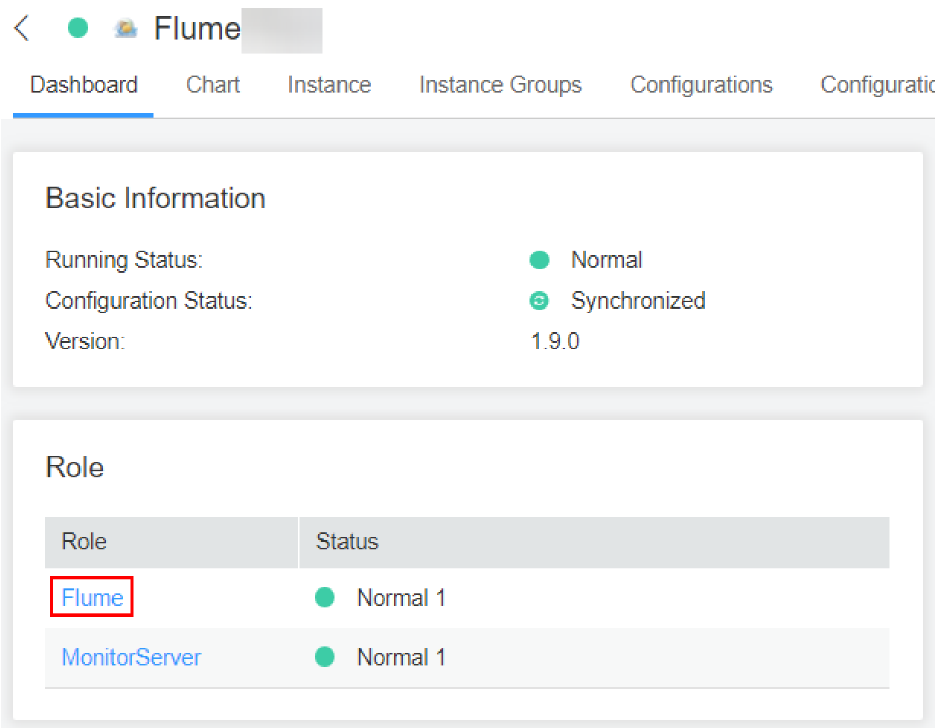
Table 7-9 Parameters to be modified for the Flume role server

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.

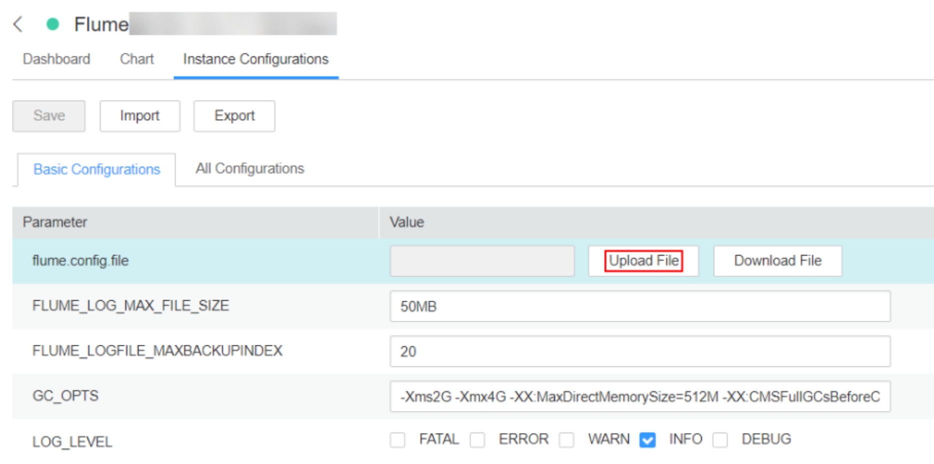
2. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Flume**. On the **Instances** tab page, click **Flume**.

Figure 7-7 Clicking the Flume role



3. Select the Flume role of the node where the configuration file is to be uploaded, choose **Instance Configurations** > **Import** beside the **flume.config.file**, and select the **properties.properties** file.

Figure 7-8 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

Scenario

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.

This section applies to MRS 3.*x* or later.

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.

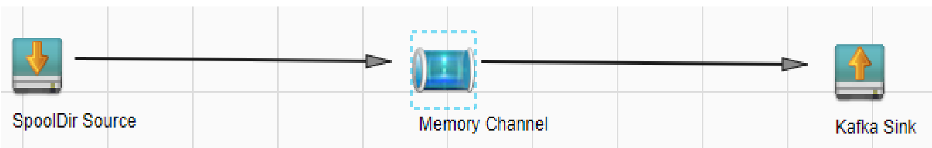
1. Log in to FusionInsight Manager. Choose **Cluster** > **Services** > **Flume** > **Configuration Tool**.

   Figure 7-9 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 7-10 Example for the Flume configuration tool

3. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-10** 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 7-10 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

Scenario

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.

This section applies to MRS 3.*x* or later.

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.

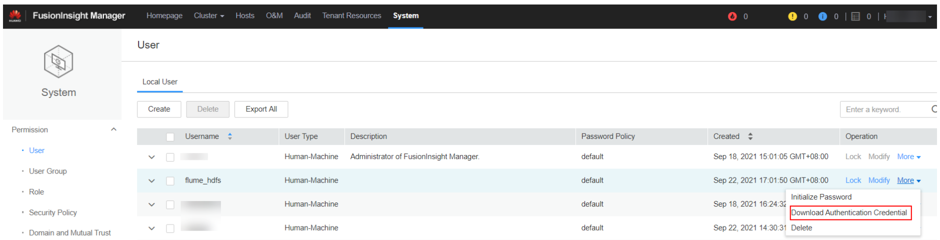
Prerequisites

- 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 7-11 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**.

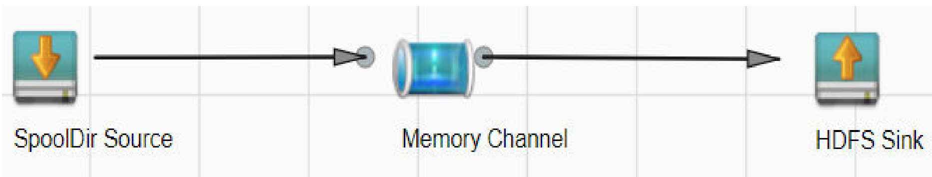
   Figure 7-12 Choosing **Configuration Tool**

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 7-13 Example for the Flume configuration tool

3. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-11** 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 7-11 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.

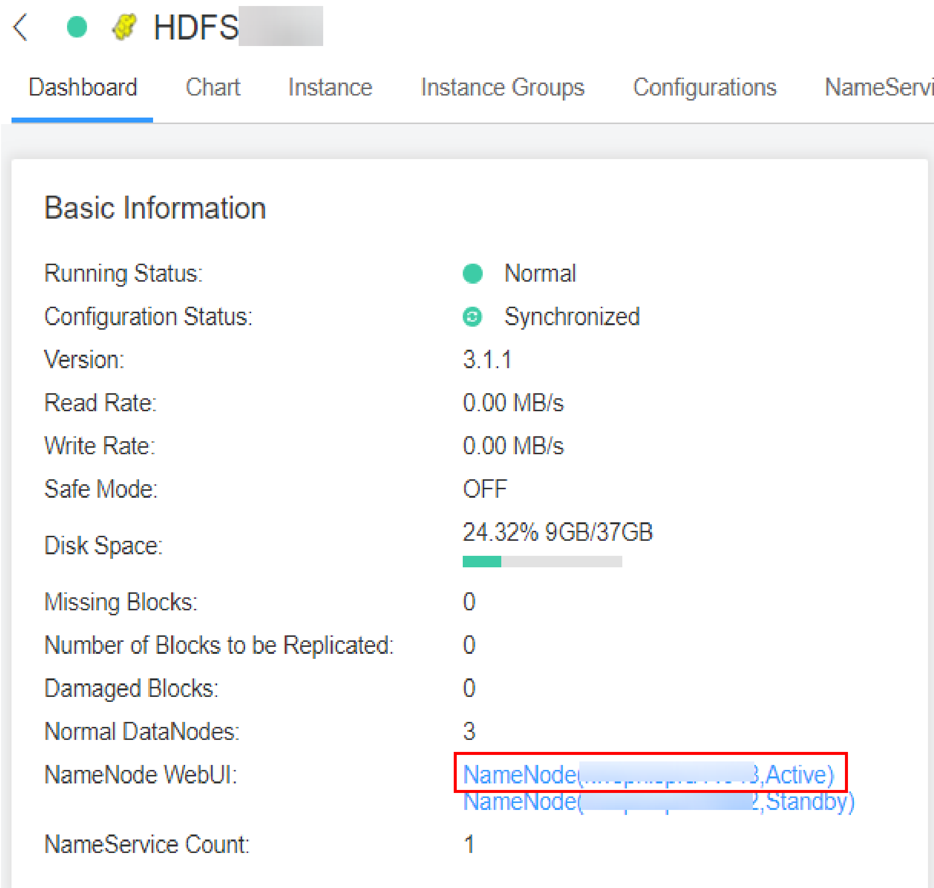
Step 3 Upload the configuration file.

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.

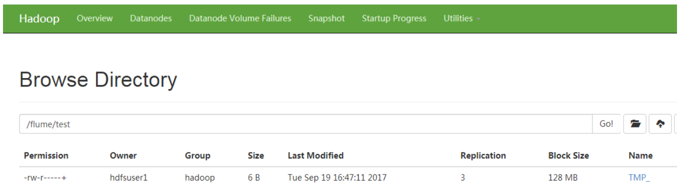
1. Log in to FusionInsight Manager as a user who has the management permission on HDFS. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > **Services** > **HDFS**. On the page that is displayed, click the **NameNode(***Node name***,Active)** link next to **NameNode WebUI** to go to the HDFS web UI. On the displayed page, choose **Utilities** > **Browse the file system**.

Figure 7-14 Accessing the HDFS WebUI



2. Check whether the data is generated in the **/flume/test** directory on the HDFS.

Figure 7-15 Checking HDFS directories and files



----End

7.5.4 Using Flume Server to Collect Dynamic Logs from Local Host to HDFS

Scenario

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.

This section applies to MRS 3.*x* or later.

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.

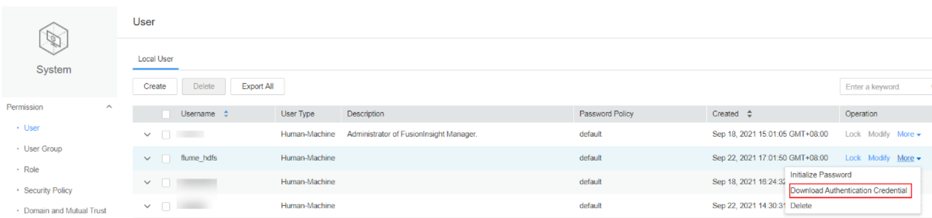
Prerequisites

- 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 7-16 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 7-17 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 7-18 Example for the Flume configuration tool

3. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-12** 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 7-12 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.

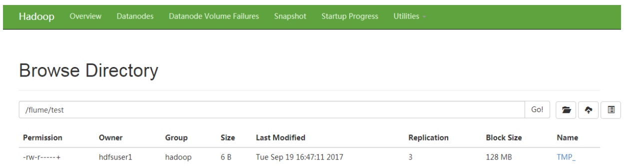
Step 3 Upload the configuration file.

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 (MRS 3.x or Later)**. 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 7-19 Checking HDFS directories and files



----End

7.5.5 Using Flume Server to Collect Logs from Kafka to HDFS

Scenario

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.

This section applies to MRS 3.*x* or later.

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.

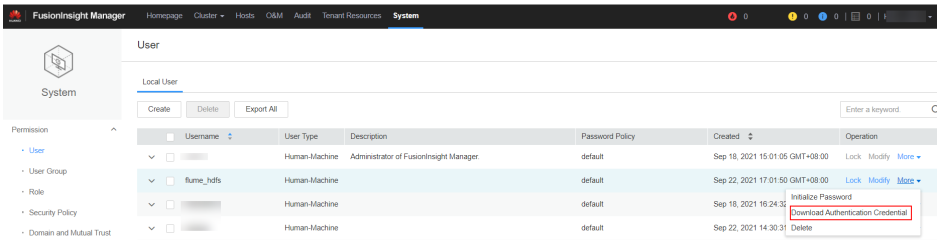
Prerequisites

- 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 7-20 Downloading the authentication credential

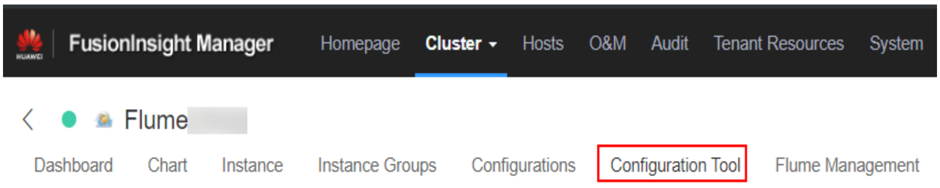


Step 2 Configure the server parameters of the Flume role.

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 7-21 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 7-22 Example for the Flume configuration tool

3. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-13** 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 7-13 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.

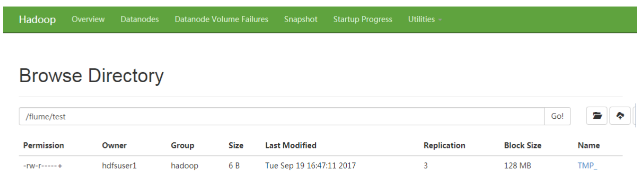
Step 3 Upload the configuration file.

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 (MRS 3.x or Later)**. 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 7-23 Checking HDFS directories and files



----End

7.5.6 Using Flume Client to Collect Logs from Kafka to HDFS

Scenario

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.

This section applies to MRS 3.*x* or later.

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**.

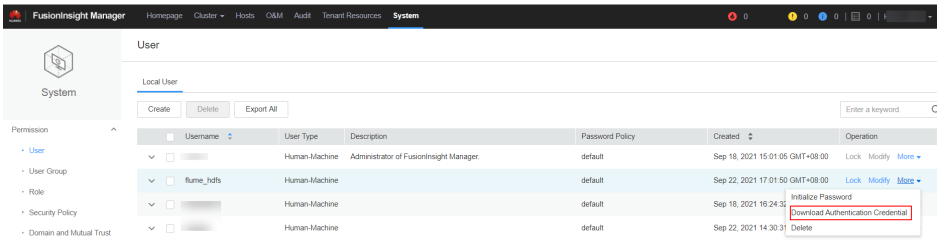
Prerequisites

- 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.

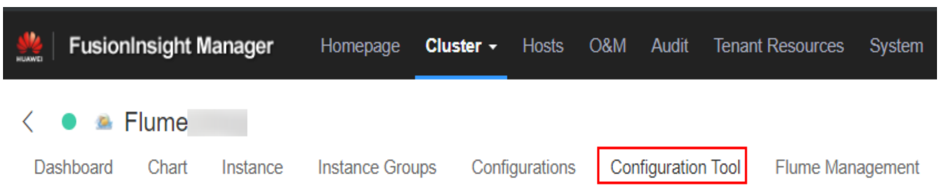
Figure 7-24 Downloading the authentication credential



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.

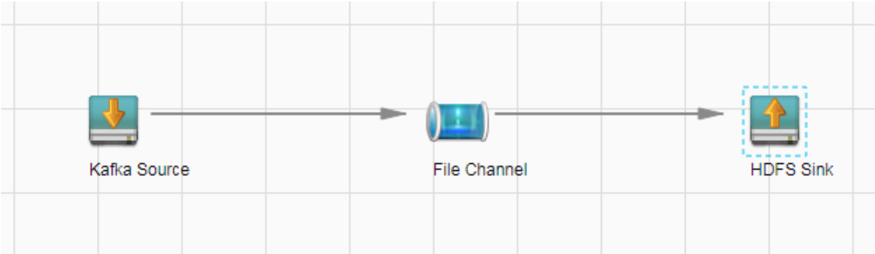
   Figure 7-25 Choosing **Configuration Tool**



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 7-26**.

Figure 7-26 Example for the Flume configuration tool



c. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-14** 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 7-14 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. This parameter is mandatory when the Flume client is used.	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/**.

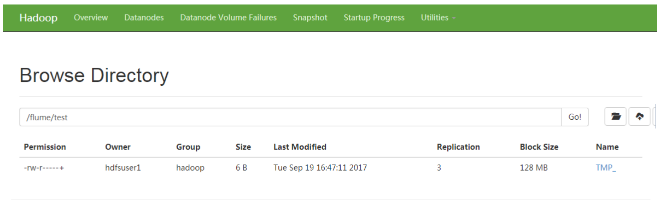
4. Go to **fusioninsight-flume-1.9.0/bin** in the installation directory of the client node and run the following command to restart the Flume process:

   ./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 (MRS 3.x or Later)**. 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 7-27 Checking HDFS directories and files



----End

7.5.7 Using Cascaded Agents to Collect Static Logs from Local Host to HBase

Scenario

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.

This section applies to MRS 3.*x* or later.

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

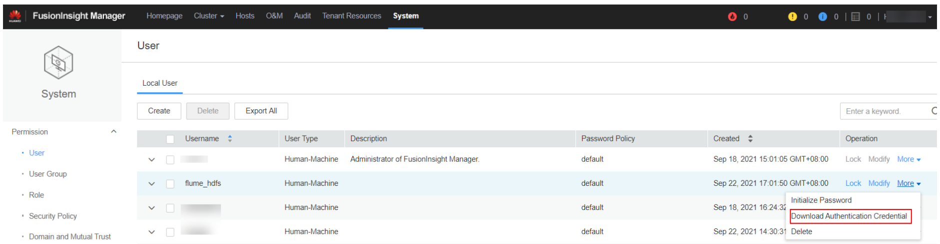
- The cluster has been installed, including the HBase and Flume services.
- The Flume client has been installed.
- The network environment of the cluster is secure.

- 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 7-28 Downloading the authentication credential

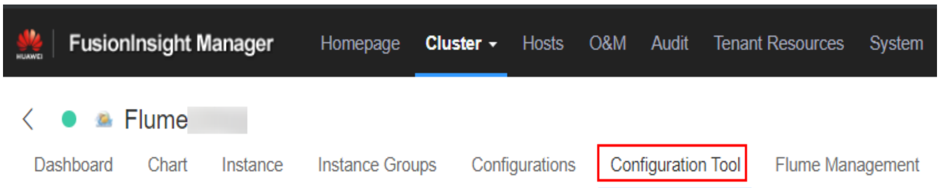


Step 2 Configure the client parameters of the Flume role.

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.

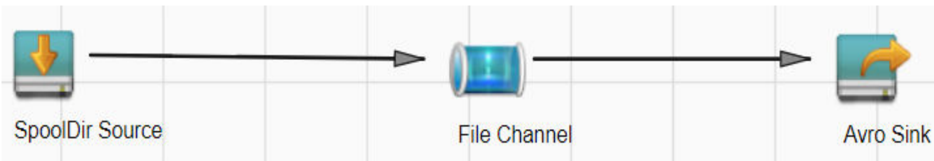
   Figure 7-29 Choosing **Configuration Tool**

   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 7-30 Example for the Flume configuration tool

   c. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-15** 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 7-15 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 configured as the connected Avro Source monitoring 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

d. Click **Export** to save the **properties.properties** configuration file to the local.

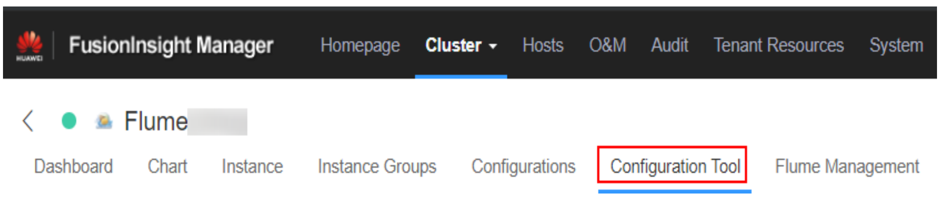
2. Upload the **properties.properties** file to **flume/conf/** under the installation directory of the Flume client.

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 7-31 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 7-32 Example for the Flume configuration tool

c. Double-click the source, channel, and sink. Set corresponding configuration parameters by referring to **Table 7-16** 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 7-16 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 port that Avro Source monitors. This parameter cannot be left blank. Set this parameter to a port that is not in use.	21154

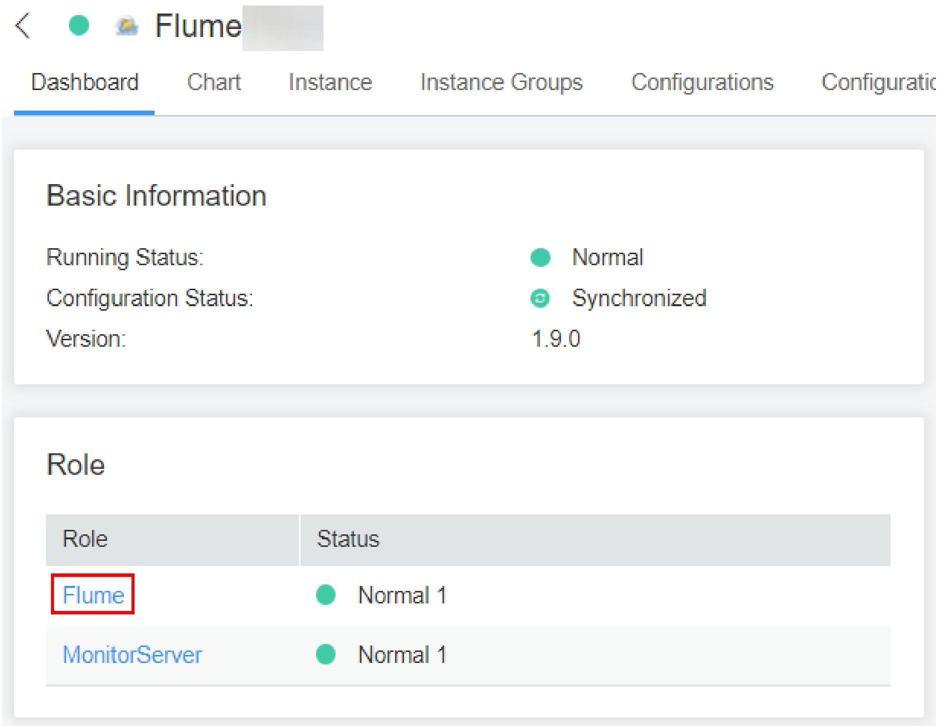
Parameter	Description	Example Value
ssl	Specifies whether to enable the SSL authentication. (You are advised to enable this function to ensure security.) Only Sources of the Avro type have this configuration item. ▪ **true** indicates that the function is enabled. ▪ **false** indicates that the client 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.

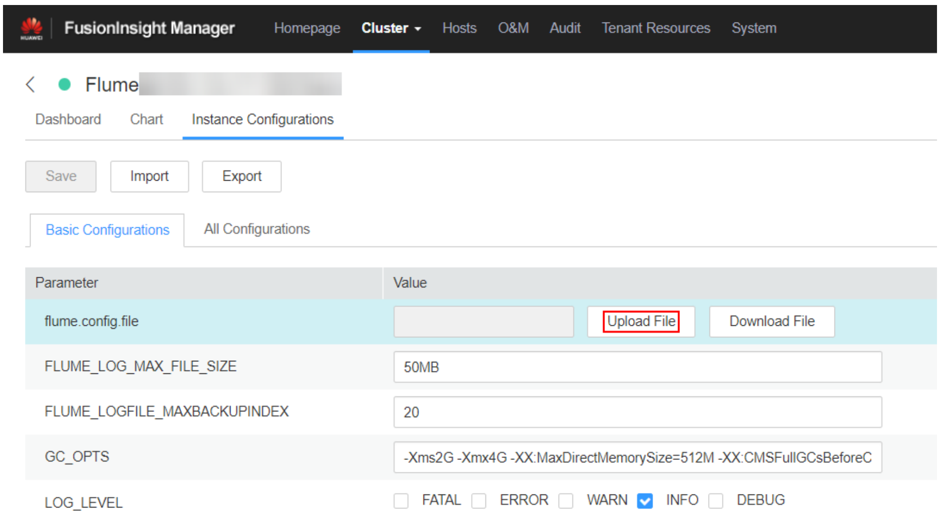
2. Log in to FusionInsight Manager and choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume**. On the displayed page, click the **Flume** role on the **Instance** tab page.

Figure 7-33 Clicking the Flume role



3. Select the Flume role of the node where the configuration file is to be uploaded, choose **Instance Configurations** > **Import** beside the **flume.config.file**, and select the **properties.properties** file.

Figure 7-34 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 Configuring the Encrypted Transmission

Scenario

This topic describes how to configure the server and client parameters of the Flume service (including the Flume role) after the cluster is installed to ensure proper service running.

This section applies to MRS 3.*x* or later.

Prerequisites

The cluster and Flume service have been installed.

Procedure

Step 1 Generate the certificate trust lists of the server and client of the Flume role respectively.

1. Remotely log in to the node using ECM where the Flume server is to be installed as user **omm**. Go to the **${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin** directory.

cd ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin

NOTE

The version 8.1.0.1 is used as an example. Replace it with the actual version number.

2. Run the following command to generate and export the server and client certificates of the Flume role:

   sh geneJKS.sh -f *xxx* **-g** *xxx*

   The generated certificate is saved in the **${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/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 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 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.

NOTE

All user-defined passwords involved in this section must meet the following strength requirements:

– The password must contain at least four types of uppercase letters, lowercase letters, digits, and special characters.
– The password must contain 8 to 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 Configure the server parameters of the Flume role and upload the configuration file to the cluster.

1. Remotely log in to any node where the Flume role is located as user **omm** using ECM. Run the following command to go to the ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin directory:

   cd ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/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.

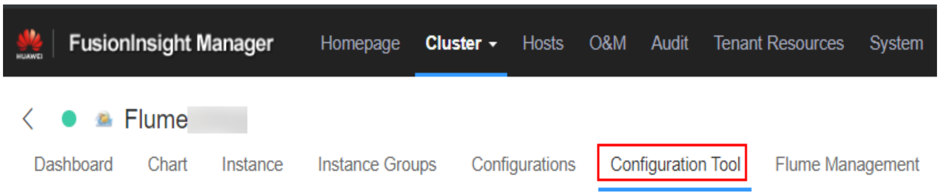
   ./genPwFile.sh

   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. Choose **Services** > **Flume** > **Configuration Tool**.

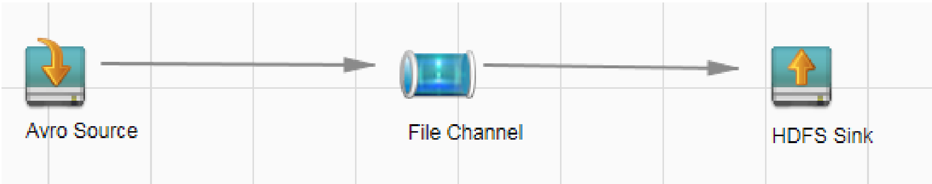
Figure 7-35 Choosing **Configuration Tool**



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 Avro Source, File Channel, and HDFS Sink, as shown in **Figure 7-36**.

Figure 7-36 Example for the Flume configuration tool



c. Double-click the source, channel, and sink. Set corresponding configuration parameters by seeing **Table 7-17** based on the actual environment.

NOTE

- If the server parameters of the Flume role have been configured, you can choose **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 **Services** > **Flume** > **Import** to change the relevant configuration items of encrypted transmission after the file is imported.
- 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.

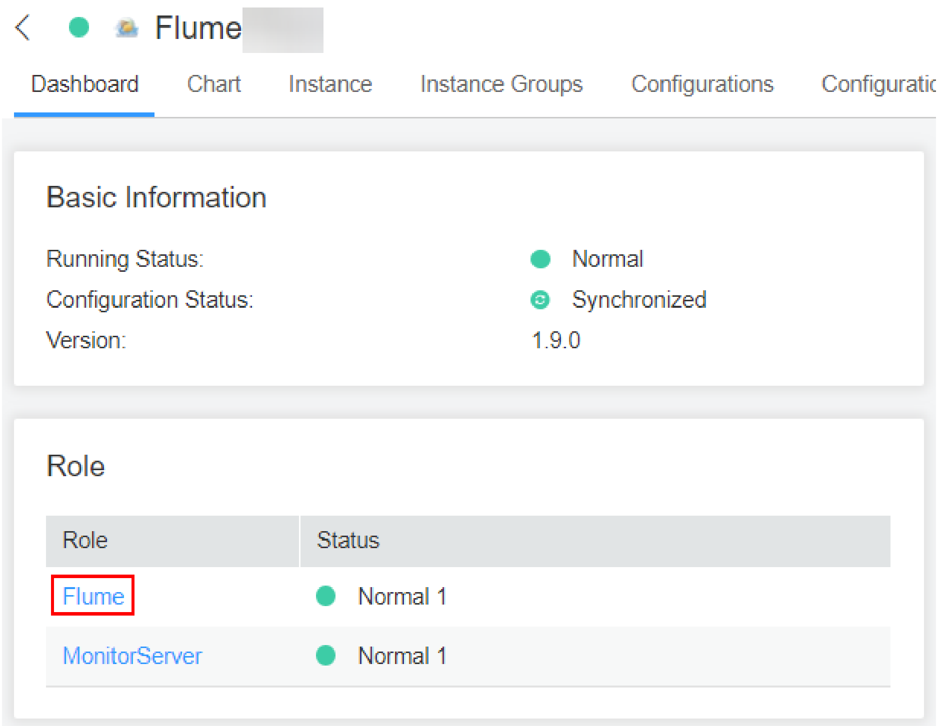
d. Click **Export** to save the **properties.properties** configuration file to the local. There can be security risks if a configuration file contains the authentication password. You are advised to delete the configuration file or use other secure methods to keep the password.

Table 7-17 Parameters to be modified of 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.) ▪ **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_8.1.0.1/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 2.2**.	-
truststore	Indicates the SSL certificate trust list of the server.	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/flume_sChatt.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 2.2**.	-

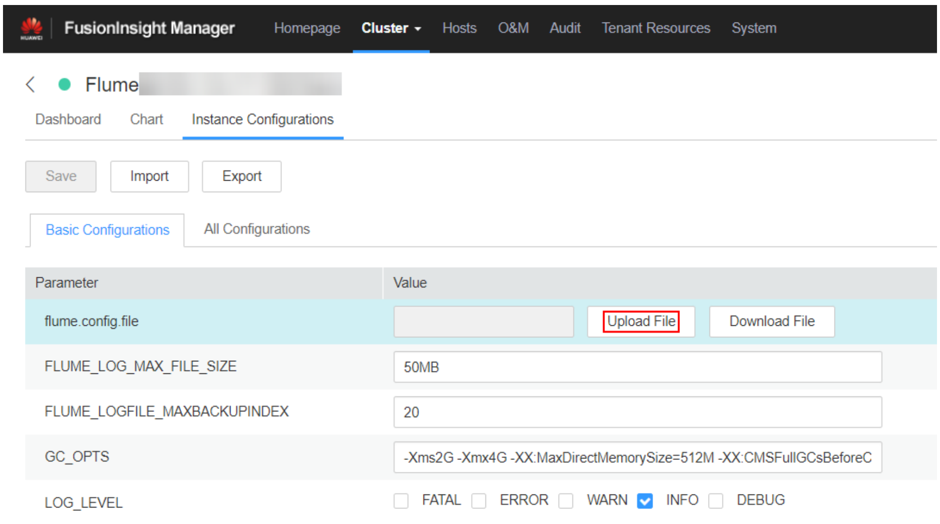
4. Log in to FusionInsight Manager and choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume**. On the displayed page, click the **Flume** role under **Role**.

Figure 7-37 Clicking the Flume role



5. Select the Flume role of the node where the configuration file is to be uploaded, choose **Instance Configurations** > **Import** beside the **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**. Click **Finish**.

Step 3 Set 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_8.1.0.1/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_8.1.0.1/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=*JDK path*** command.

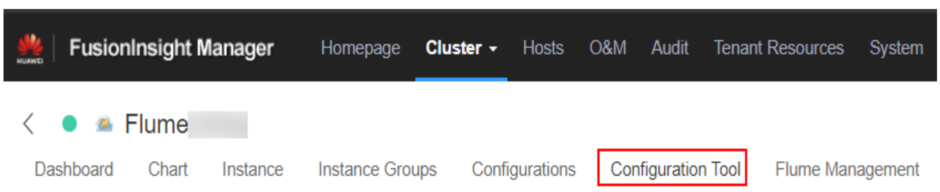
   ```
   JAVA_HOME is null in current user,please install the JDK and set the JAVA_HOME
   ```

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 3.5**.

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**.

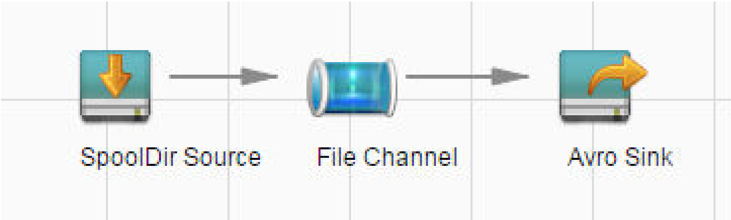
   Figure 7-39 Choosing **Configuration Tool**

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.

For example, use SpoolDir Source, File Channel, and Avro Sink, as shown in **Figure 7-40**.

Figure 7-40 Example for the Flume configuration tool



c. Double-click the source, channel, and sink. Set corresponding configuration parameters by seeing **Table 7-18** 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.
- 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 7-18 Parameters to be modified of the Flume role client

Parameter	Description	Example Value
ssl	Indicates whether to enable the SSL authentication. (You are advised to enable this function to ensure security.) ■ **true** indicates that the function is enabled. ■ **false** indicates that the client authentication function is not enabled.	true
keystore	Specified the client certificate.	/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 3.3**.	-
truststore	Indicates the SSL certificate trust list of the client.	/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 3.3**.	-

6. Upload the **properties.properties** file to **flume/conf/** under the installation directory of the Flume client.

----End

7.6.2 Using Cascaded Agents to Collect Static Logs from Local Host to HDFS

Scenario

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.

This section applies to MRS 3.*x* or later.

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_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin** directory.

   cd ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/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 *Password* **-g** *Password*

   The generated certificate is saved in the **${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/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 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 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.

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_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin directory:

   cd ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/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.

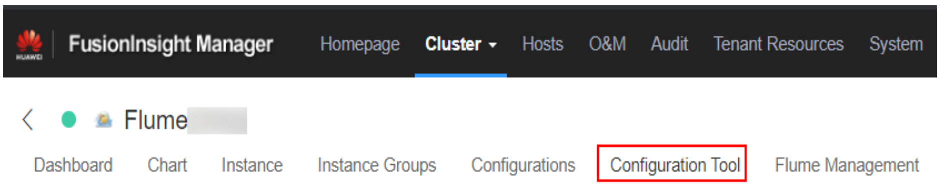
   ./genPwFile.sh

   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 7-41 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 7-42 Example for the Flume configuration tool

c. Double-click the source, channel, and sink. Set corresponding configuration parameters by seeing **Table 7-19** 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. 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.

Table 7-19 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
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 port that Avro Source monitors. This parameter cannot be left blank. Set this parameter to a port that is not in use.	21154

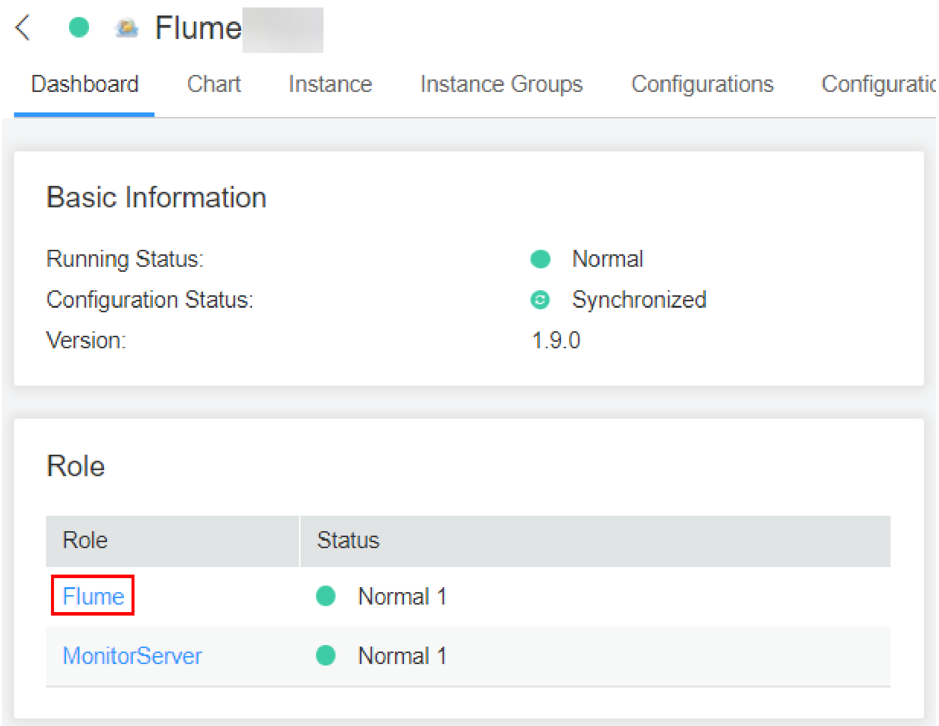
Parameter	Description	Example Value
ssl	Indicates whether to enable the SSL authentication. (You are advised to enable this function to ensure security.) Only Sources of the Avro type have this configuration item. ■ **true** indicates that the function is enabled. ■ **false** indicates that the client authentication function is not enabled.	true
keystore	Indicates the server certificate.	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/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_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/flume_sChatt.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 3.2**.	-

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/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_
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

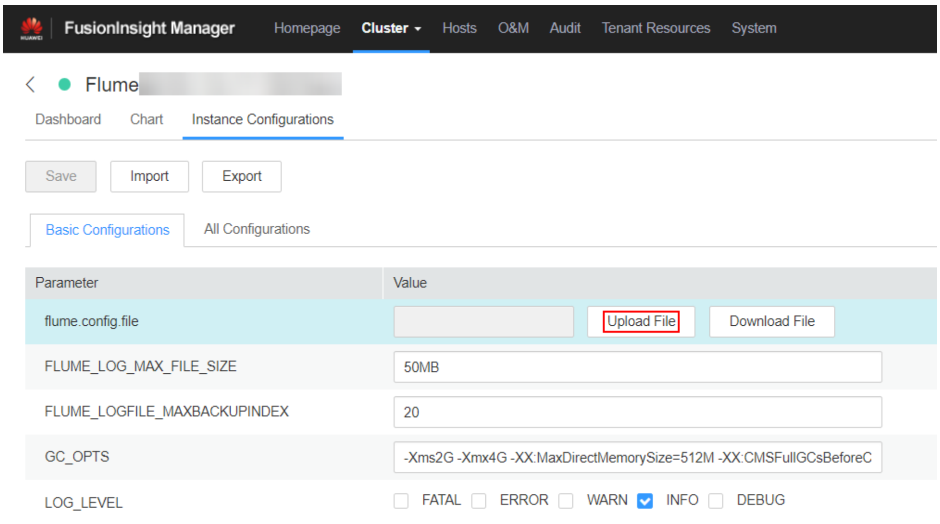
4. Log in to FusionInsight Manager and choose **Cluster** > *Name of the desired cluster* > **Services** > **Flume**. On the displayed page, click the **Flume** role under **Role**.

Figure 7-43 Clicking the Flume role



5. Select the Flume role of the node where the configuration file is to be uploaded, choose **Instance Configurations** > **Import** beside the **flume.config.file**, and select the **properties.properties** file.

Figure 7-44 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_8.1.0.1/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_8.1.0.1/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.

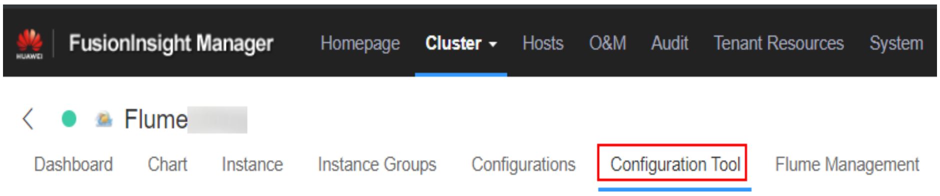
   ```
   JAVA_HOME is null in current user,please install the JDK and set the JAVA_HOME
   ```

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**.

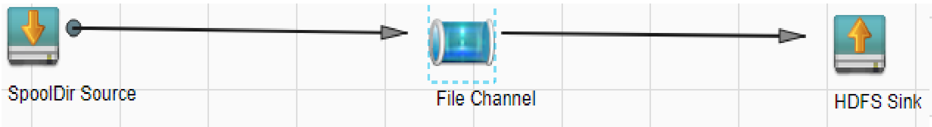
Figure 7-45 Choosing **Configuration Tool**



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 7-46 Example for the Flume configuration tool



c. Double-click the source, channel, and sink. Set corresponding configuration parameters by seeing **Table 7-20** 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 7-20 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

Parameter	Description	Example Value
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
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

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
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

Parameter	Description	Example Value
port	Specifies the port that Avro Sink monitors. This parameter cannot be left blank. It must be configured as the connected Avro Source monitoring port.	21154
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

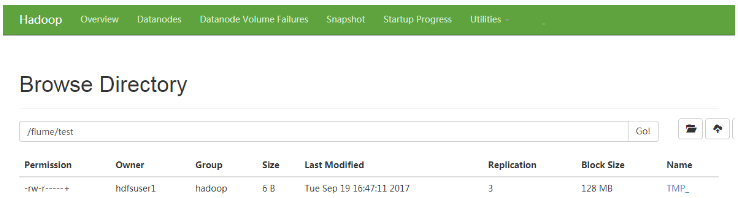
Parameter	Description	Example Value
truststore-password	Specifies the trust list password, which is the password required to obtain the truststore information. Enter the value of **password** obtained in **Step 4.3**.	-

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 (MRS 3.x or Later)**. 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 7-47 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.

- For versions earlier than MRS 3.x, the encryption path is **/opt/Bigdata/MRS_XXX/install/FusionInsight-Flume-*Flume component version number*/flume/bin/genPwFile.sh**.
- For MRS 3.*x* or later, the encryption path is **/opt/Bigdata/FusionInsight_Porter_XXX/install/FusionInsight-Flume-*Flume component version number*/flume/bin/genPwFile.sh**. *XXX* indicates the product version number.

----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 Typical Flume Configuration Parameters

For versions earlier than MRS 3.x, configure Flume parameters in the **properties.properties** file.

For MRS 3.*x* or later, some parameters can be configured on Manager.

Overview

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**

  Avro Source monitors the Avro port, receives data from the external Avro client, and puts the data into the configured channel. **Table 7-21** lists common configurations.

Figure 7-48 Avro Source

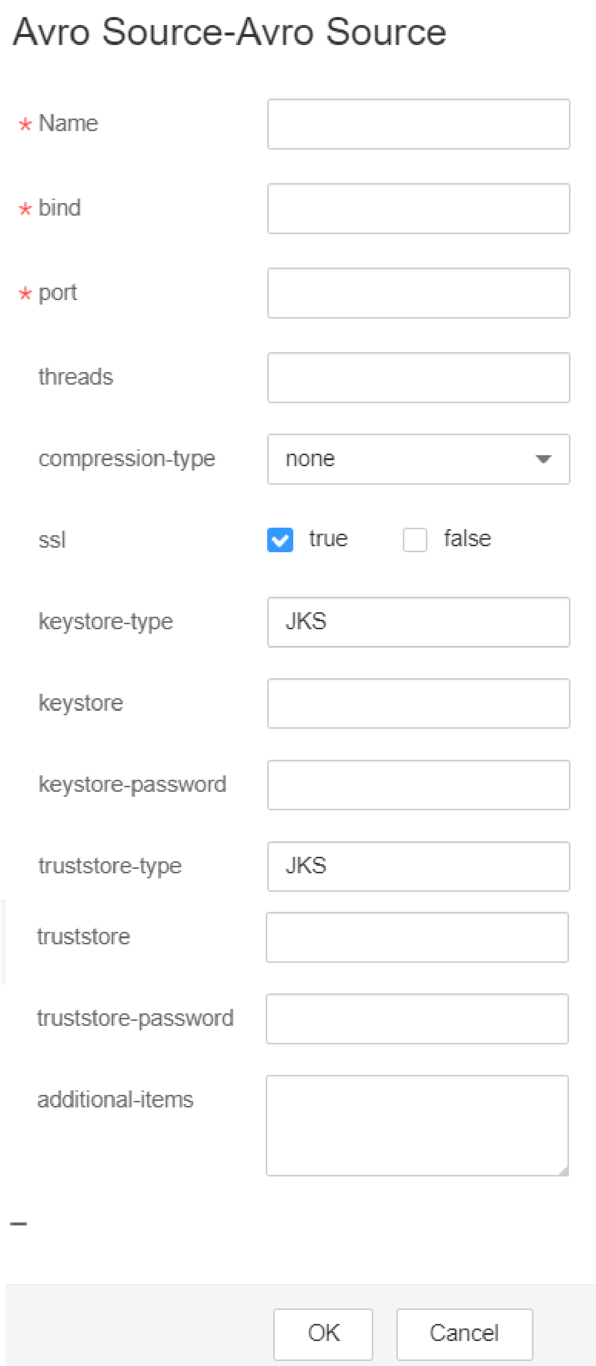


Table 7-21 Common configurations of an Avro source

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 7-49 SpoolDir Source

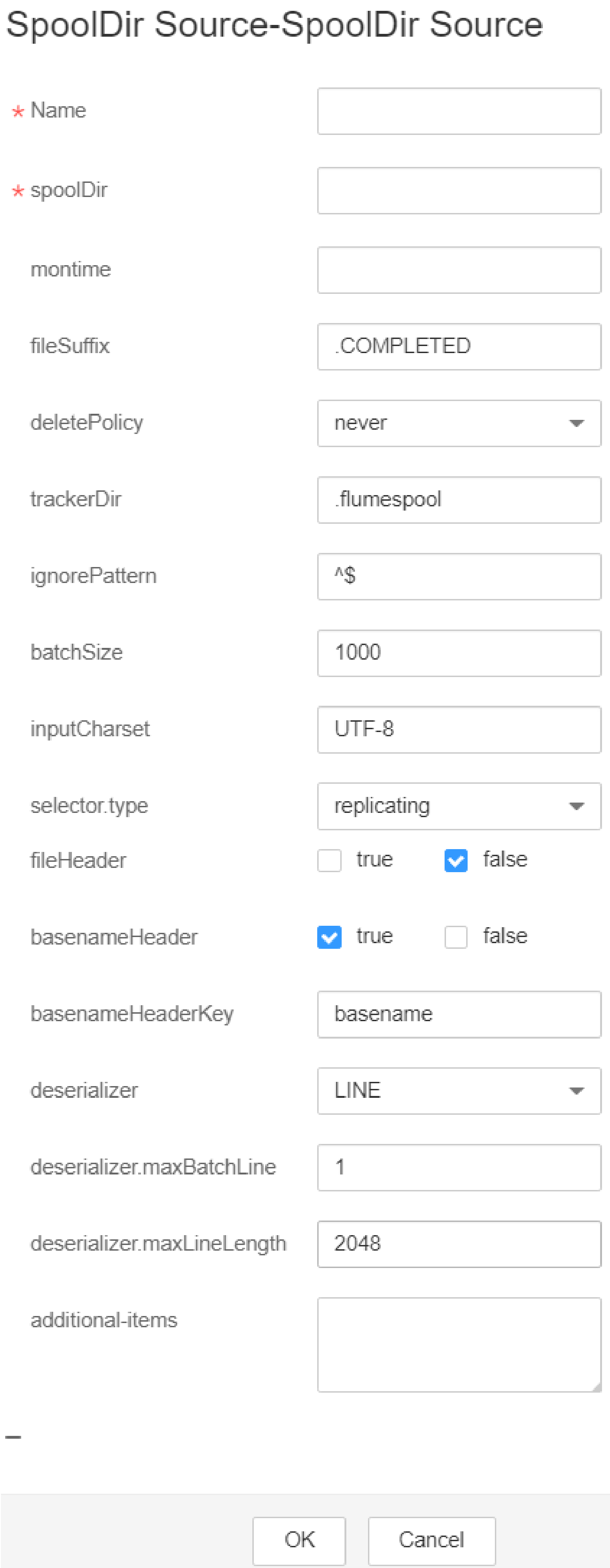


Table 7-22 Common configurations of a SpoolDir source

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 7-50 Kafka Source

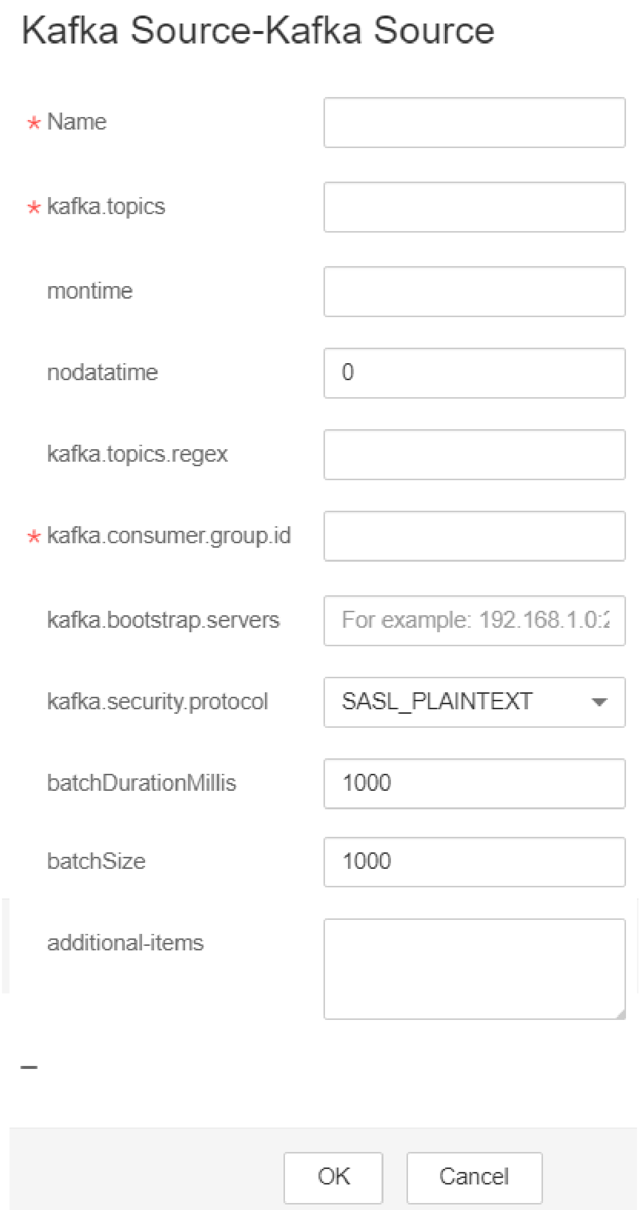


Table 7-23 Common configurations of a Kafka source

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 7-24** lists common configurations.

Figure 7-51 Taildir Source

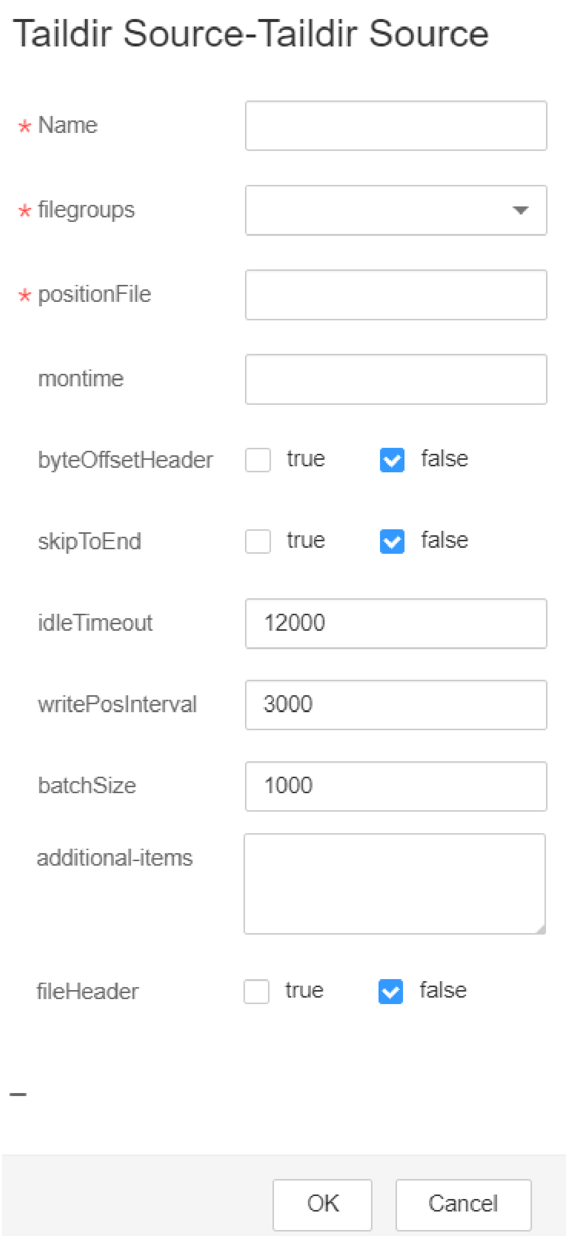


Table 7-24 Common configurations of a Taildir source

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 7-25** lists common configurations.

Figure 7-52 Http Source

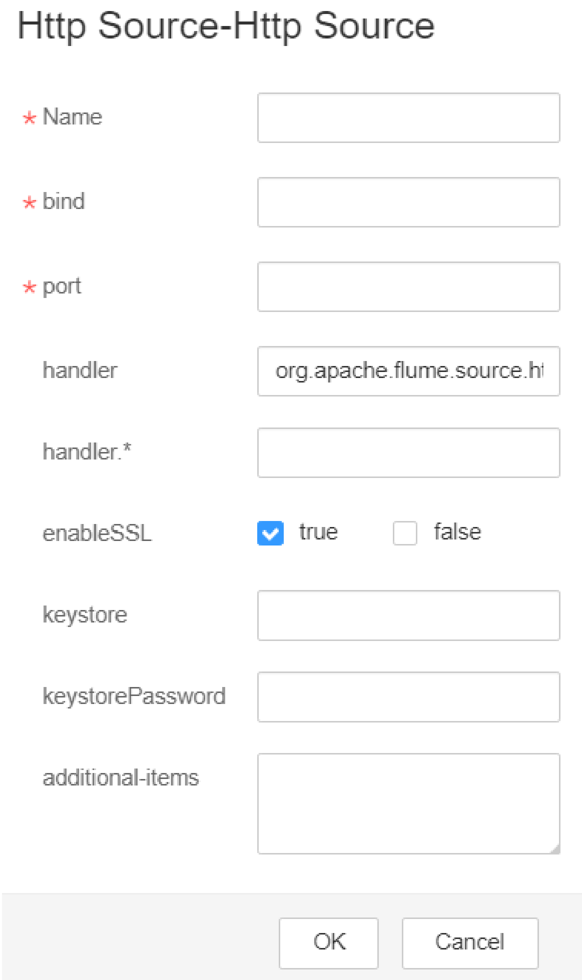


Table 7-25 Common configurations of an HTTP source

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 7-26** lists common configurations.

Figure 7-53 Memory Channel

Table 7-26 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 7-27** lists common configurations.

Figure 7-54 File Channel

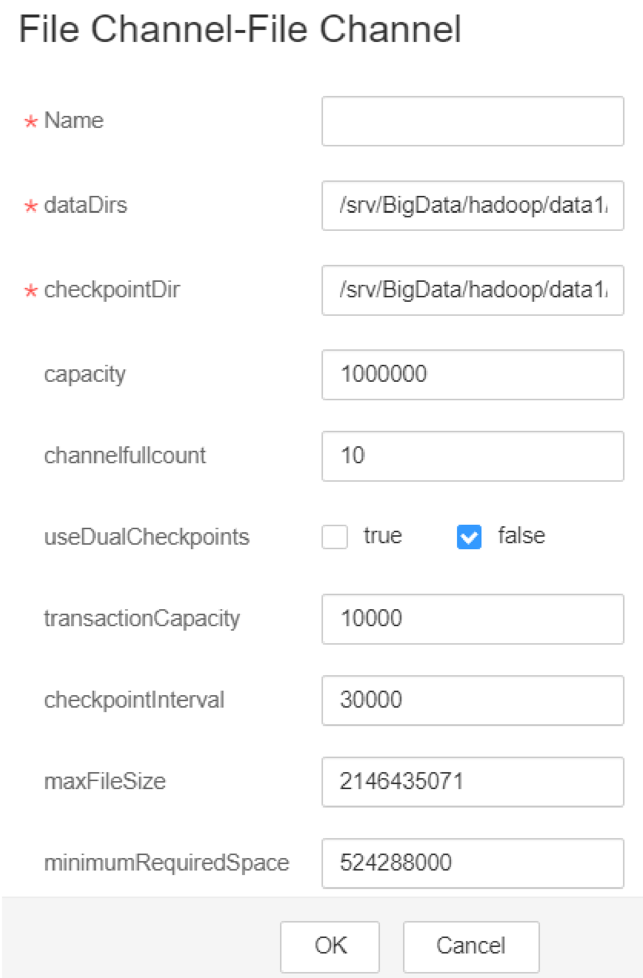


Table 7-27 Common configurations of a file channel

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 7-55 Kafka Channel

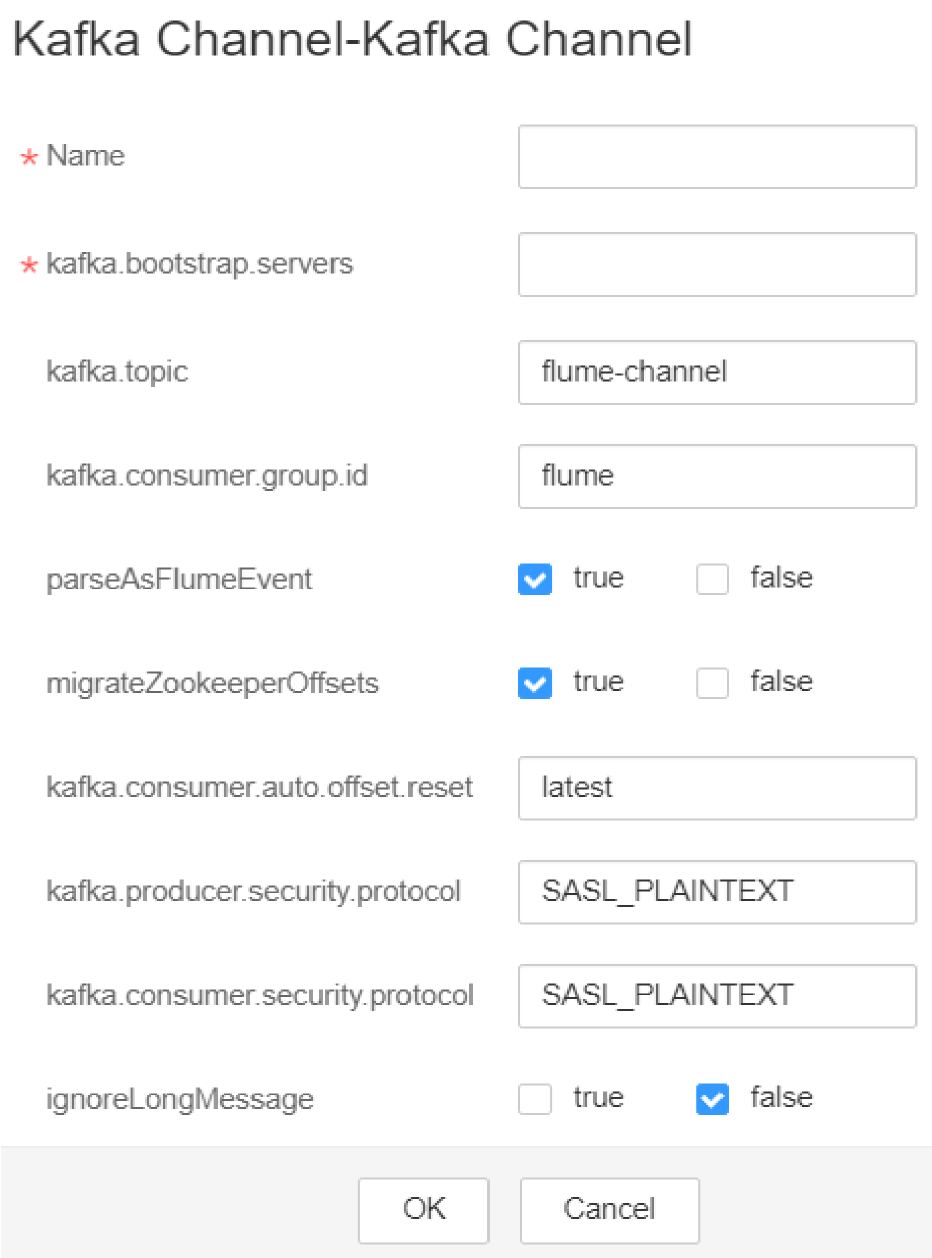


Table 7-28 Common configurations of a Kafka channel

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 7-29** lists common configurations.

Figure 7-56 HDFS Sink

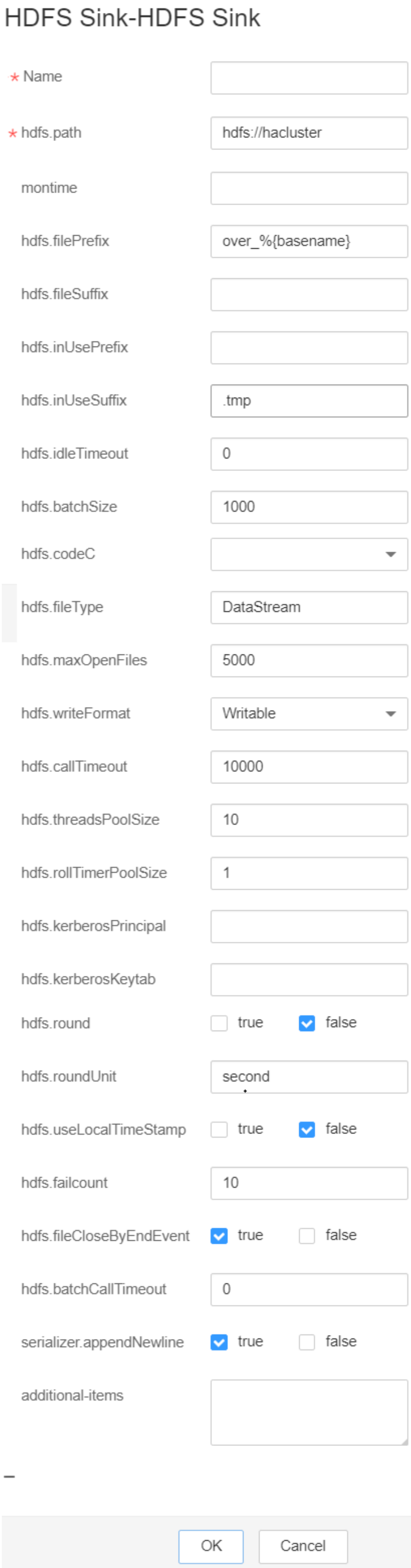


Table 7-29 Common configurations of an HDFS sink

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 file rolling by time, in seconds. Set **hdfs.fileCloseByEndEvent** to **false** if you set this parameter.
hdfs.rollSize	1024	Specifies file rolling by size, in bytes. Set **hdfs.fileCloseByEndEvent** to **false** if you set this parameter.
hdfs.rollCount	10	Specifies file rolling by the number of events, 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**

  Avro Sink converts events into Avro events and sends them to the monitoring ports of the hosts. **Table 7-30** lists common configurations.

Figure 7-57 Avro Sink

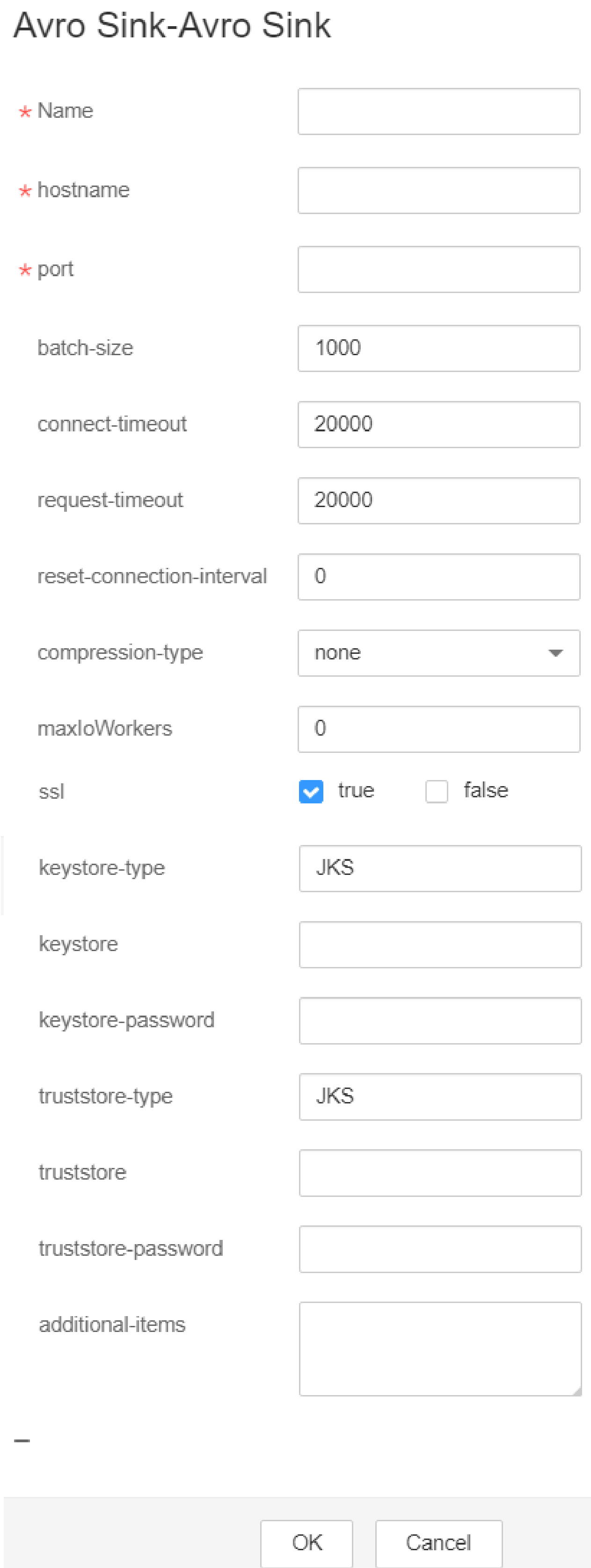


Table 7-30 Common configurations of an Avro sink

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 7-31** lists common configurations.

Figure 7-58 HBase Sink

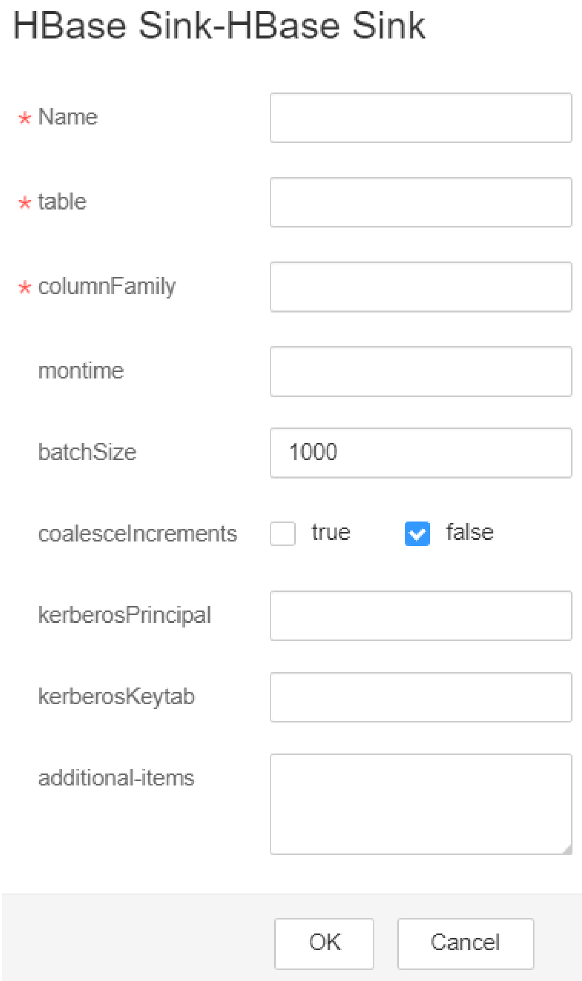


Table 7-31 Common configurations of an HBase sink

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 7-32** lists common configurations.

Figure 7-59 Kafka Sink

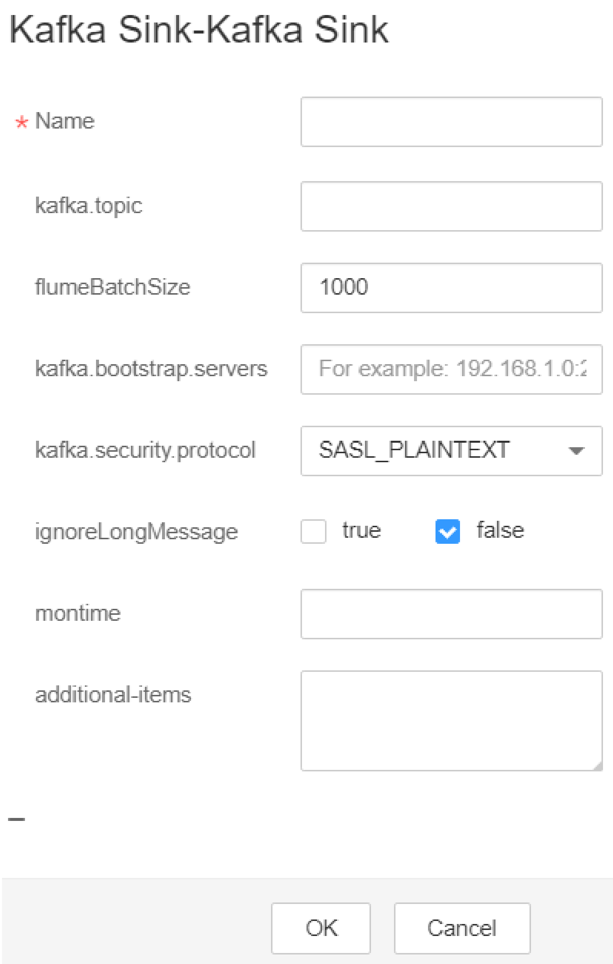


Table 7-32 Common configurations of a Kafka sink

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 Service Configuration Guide

This section applies to MRS 3.*x* or later.

This configuration guide describes how to configure common Flume services.

NOTE

- Parameters in bold in the following tables are mandatory.
- 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**

  Avro Source monitors the Avro port, receives data from the external Avro client, and puts the data into the configured channel. Common configurations are as follows:

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.
type	avro	Specifies the type of the avro source, which must be **avro**.
bind	-	Specifies the monitoring host name and IP address.
port	-	Specifies the bound monitoring port. Ensure that this port is not occupied.
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.

Parameter	Default Value	Description
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.
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.

Parameter	Default Value	Description
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.
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 7-34 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.
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.

Parameter	Default Value	Description
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.
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.

Parameter	Default Value	Description
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.
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**.

Parameter	Default Value	Description
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 7-35 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**.

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. 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**.
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.

Parameter	Default Value	Description
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.
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).

Parameter	Default Value	Description
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 7-36 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.
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.

Parameter	Default Value	Description
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.
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 7-37 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 monitoring host name and IP address.
port	-	Specifies the bound monitoring port. Ensure that this port is not occupied.
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.

Parameter	Default Value	Description
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 monitoring host name and IP address.
port	-	Specifies the bound monitoring 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 machine-machine account must be used. 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.

Parameter	Default Value	Description
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.

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 7-38 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**.

Parameter	Default Value	Description
channelfullcount	10	Specifies the channel full count. When the count reaches the threshold, an alarm is reported.
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 7-39 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.

Parameter	Default Value	Description
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.
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.

Parameter	Default Value	Description
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.
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 7-40 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.

Parameter	Default Value	Description
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**.
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

Parameter	Default Value	Description
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 7-41 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**.

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. 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.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.

Parameter	Default Value	Description
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.
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

Parameter	Default Value	Description
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 7-42 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 file rolling by time, in seconds. Set **hdfs.fileCloseByEndEvent** to **false** if you set this parameter.
hdfs.rollSize	1024	Specifies file rolling by size, in bytes. Set **hdfs.fileCloseByEndEvent** to **false** if you set this parameter.

Parameter	Default Value	Description
hdfs.rollCount	10	Specifies file rolling by the number of events, 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.
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.

Parameter	Default Value	Description
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.
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.

Parameter	Default Value	Description
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**.
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**

  Avro Sink converts events into Avro events and sends them to the monitoring ports of the hosts. Common configurations are as follows:

Table 7-43 Common configurations of an Avro sink

Parameter	Default Value	Description
channel	-	Specifies the channel connected to the sink.

Parameter	Default Value	Description
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 monitoring port. Ensure that this port is not occupied.
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.

Parameter	Default Value	Description
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.
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.

Parameter	Default Value	Description
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.
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 7-44 Common configurations of an HBase sink

Parameter	Default Value	Description
channel	-	Specifies the channel connected to the sink.

Parameter	Default Value	Description
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
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 7-45 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.
flumeBatchSize	1000	Specifies the number of events written into Kafka in batches.
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).

Parameter	Default Value	Description
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**

  Thrift Sink converts events to Thrift events and sends them to the monitoring port of the configured host. Common configurations are as follows:

Table 7-46 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**.

Parameter	Default Value	Description
hostname	-	Specifies the bound host name or IP address.
port	-	Specifies the bound monitoring 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.
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.

Parameter	Default Value	Description
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.
  - Configure the failover and load_balance mechanisms for Sink Processor. The following shows a load balancing example.
    ```
    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.

7.8.3 Flume Logs 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 7-47 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

Log Level

Table 7-48 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 7-48 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 7-49 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

7.8.4 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

7.8.5 Viewing Flume Client Monitoring Information

Scenario

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.

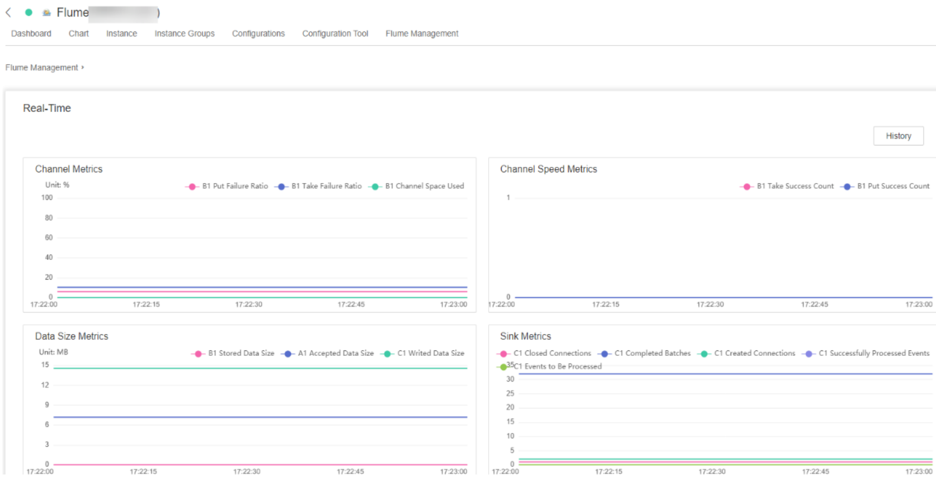
This section applies to MRS 3.*x* or later.

Procedure

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 7-60 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

7.8.6 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

This section describes how to use environment variables in the **properties.properties** configuration file.

This section applies to MRS 3.*x* or later.

Step 1 Install the Flume client.

Step 2 Log in to the node where the Flume client is installed as user **root**.

Step 3 Switch to the following directory:

cd *Flume client installation directory*/**fusioninsight-flume**-*Flume component version*/**conf**

Step 4 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 5 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 6 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 6** 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

This section describes how to perform secondary development for third-party plug-ins.

This section applies to MRS 3.*x* or later.

Step 1 Compress the self-developed code into a JAR package.

Step 2 Install the Flume server or client, for example, in the */opt/flumeclient* directory.

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:

```
[root@[redacted] plugins.d]#mkdir thirdPlugin
[root@[redacted] plugins.d]#ll
total 8
drwxr-x--- 3 root root 4096 [redacted] native
drwxr-xr-x 2 root root 4096 [redacted] thirdPlugin
[root@[redacted] plugins.d]#cd thirdPlugin/
[root@[redacted] thirdPlugin]#mkdir lib libext native
[root@[redacted] thirdPlugin]#ll
total 12
drwxr-xr-x 2 root root 4096 [redacted] lib
drwxr-xr-x 2 root root 4096 [redacted] libext
drwxr-xr-x 2 root root 4096 [redacted] native
[root@[redacted] thirdPlugin]#
```

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.

This section applies to MRS 3.*x* or later.

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 HBase

8.1 Creating HBase Roles

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

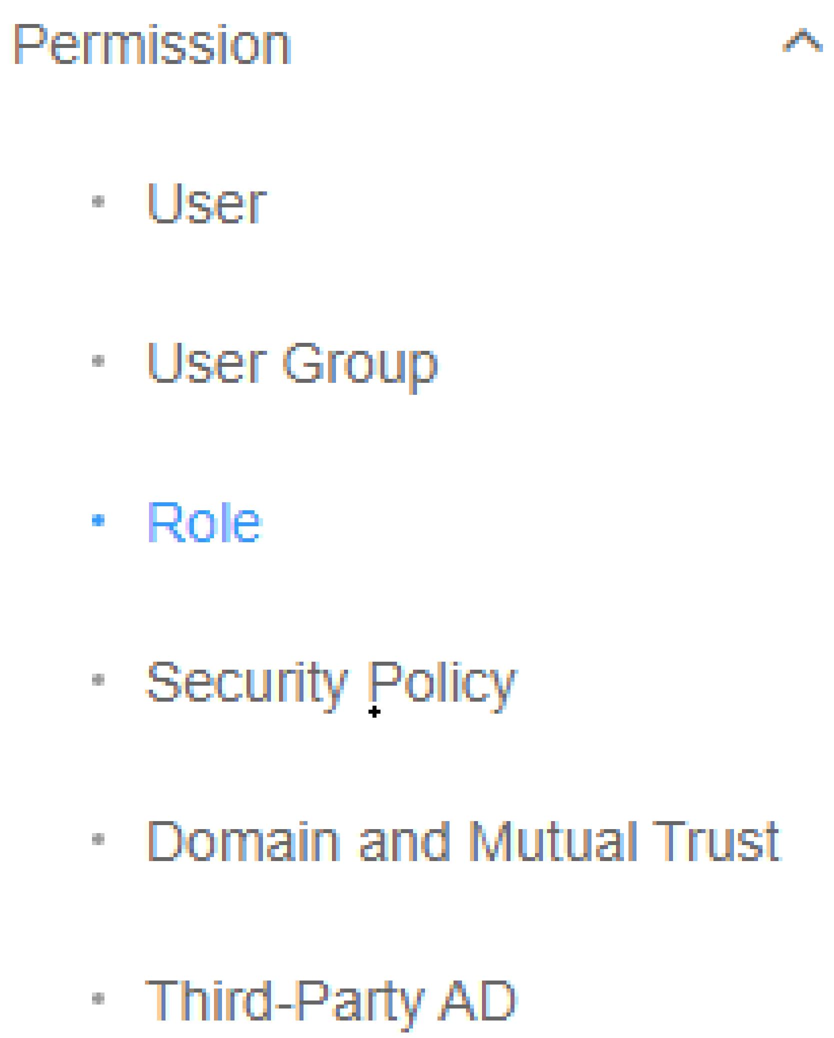
- This section applies to MRS 3.*x* or later.
- HBase roles can be created only for clusters with Kerberos authentication enabled (security mode).
- If the current component uses Ranger for permission control, you need to configure related policies based on Ranger for permission management. For details, see **Adding a Ranger Access Permission Policy for HBase**.

Prerequisites

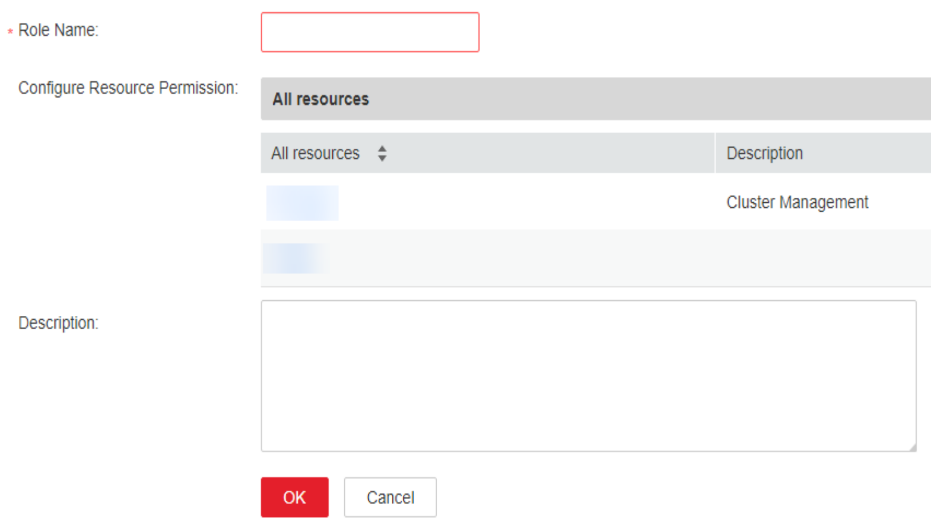
- The MRS cluster administrator has understood service requirements.
- You have logged in to Manager.

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 8-1**.

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 8-1 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

8.2 Using the HBase Client

Scenario

This section describes how to use the HBase client in an O&M scenario or a service scenario.

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 (Versions Earlier Than MRS 3.x)

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 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. If Kerberos authentication is disabled for the current cluster, skip this step.

kinit *Component service user*

For example, **kinit hbaseuser**.

Step 5 Run the following HBase client command:

hbase shell

----End

Using the HBase Client (MRS 3.*x* or Later)

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 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. If Kerberos authentication is disabled for the current cluster, skip this step.

kinit *Component service user*

For example, **kinit hbaseuser**.

Step 5 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 8-2 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. The table name and scanner must be specified, for example, **scan 'test'**.

8.3 Quickly 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. For example, how to use the client to create tables, insert data into tables, modify tables, read and delete table data, and delete tables.

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 8-3 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

For MRS clusters earlier than version 3.*x*, perform the following steps:

Step 1 Download the client configuration file.

1. Log in to MRS Manager. For details, see **Accessing Manager**. Then, choose **Services**.
2. Click **Download Client**.

   Set **Client Type** to **Only configuration files**, **Download To** to **Server**, and click **OK** to generate the client configuration file. The generated file is saved in the **/tmp/MRS-client** directory on the active management node by default. You can customize the file path.

Step 2 Log in to the active management node of MRS Manager.

1. On the **Node** tab page, view the **Name** parameter. The node that contains **master1** in its name is the Master1 node. The node that contains **master2** in its name is the Master2 node.

   The active and standby management nodes of MRS Manager are installed on Master nodes by default. Because Master1 and Master2 are switched over in active and standby mode, Master1 is not always the active management node of MRS Manager. Run a command in Master1 to check whether Master1 is active management node of MRS Manager. For details about the command, see **Step 2.4**.
2. Log in to the Master1 node using the password as user **root**.
3. Run the following commands to switch to user **omm**:

   sudo su - root

   su - omm
4. Run the following command to check the active management node of MRS Manager:

   sh ${BIGDATA_HOME}/om-0.0.1/sbin/status-oms.sh

   In the command output, the node whose **HAActive** is **active** is the active management node, and the node whose **HAActive** is **standby** is the standby management node. In the following example, **mgtomsdat-sh-3-01-1** is the active management node, and **mgtomsdat-sh-3-01-2** is the standby management node.

```
Ha mode
double
NodeName          HostName                    HAVersion        StartTime                HAActive
HAAllResOK        HARunPhase
192-168-0-30        mgtomsdat-sh-3-01-1         V100R001C01        2019-11-18 23:43:02
active            normal            Actived
192-168-0-24        mgtomsdat-sh-3-01-2         V100R001C01        2019-11-21 07:14:02
standby            normal            Deactived
```

5. Log in to the active management node, for example, **192-168-0-30** of MRS Manager 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/MRS-client/MRS_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. The current user must have the permission to create HBase tables. If Kerberos authentication is disabled for the current cluster, skip this step.

   kinit *MRS cluster user*

   For example, **kinit hbaseuser**.

4. Run the following HBase client command:

   hbase shell

Step 6 Run the following commands on the HBase client to implement service A.

1. Create the **user_info** user information table according to **Table 8-3** 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'*user_info*','*12005000201*','i'

6. Delete the user information table.

   disable'*user_info*'

   drop '*user_info*'

----End

For MRS 3.*x* or later clusters, perform the following steps:

Step 1 Use the client on the active management node.

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 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. The current user must have the permission to create HBase tables. If Kerberos authentication is disabled for the current cluster, skip this step.

   kinit *MRS cluster user*

   For example, **kinit hbaseuser**.

4. 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 8-3** 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

8.4 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**.

8.5 HBase Data Operations

8.5.1 Creating HBase Indexes for Data Query

Scenario

HBase is a distributed database for key-value pairs. HIndex enables HBase to index based on the values of certain columns to narrow down the search scope and speed up searches.

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.

Creating an HBase HIndex

Step 1 Install the HBase client. For details, see **Using the HBase Client**.

Step 2 Go 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. In normal mode, user authentication is not required.

kinit *Component service user*

Step 5 Run the following command to access HIndex:

hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer

Table 8-4 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'

Function	Command
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

8.5.2 Configuring HBase Data Compression and Encoding Formats

Scenario

HBase encodes data blocks in HFiles to reduce duplicate keys in Key-Value pairs, 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 the cluster, 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 HBase Roles**.

Configuring 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 command to log in to the HBase client:
     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.

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 command to log in to the HBase client:

hbase shell

f. Run the following command to modify the HBase table:

alter '*t1*', {NAME => '*f1*', COMPRESSION => '*SNAPPY*', DATA_BLOCK_ENCODING => '*FAST_DIFF*'}

8.6 Enterprise-Class Enhancements of HBase

8.6.1 Configuring HBase Local Secondary Indexes for Faster Queries

8.6.1.1 About 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. However, this process also scans unnecessary data, limiting its performance for 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.

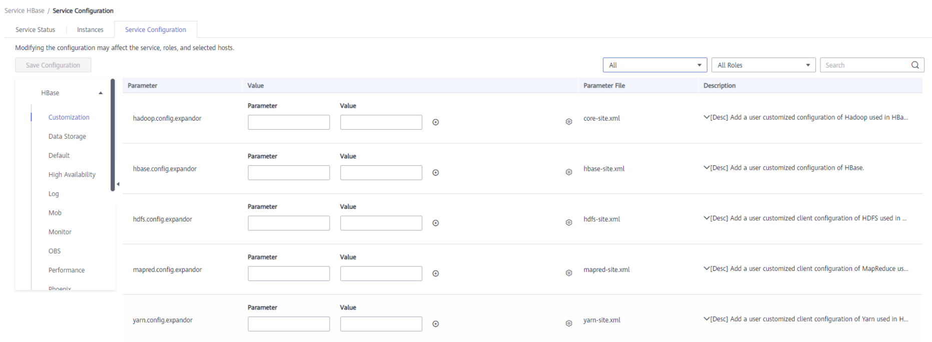
Figure 8-1 HBase HIndex

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 a Local HBase Secondary Index

1. Log in to the MRS console, click a cluster name and choose **Components**.
2. In the **Components** tab, click **HBase** and then the **Service Configuration** tab, and select **All** from the **Basic** drop-down list.



3. View parameters on the displayed page.

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.

Navigation Path	Parameter	Default Value	Description
	hbase.coprocessor.wal.classes	org.apache.hadoop.hbase.hindex.server.regionserver.HIndexRegionServerCoprocessor,org.apache.hadoop.hbase.JMXListener,org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor	This coprocessor is used for Replication, which filters index data to prevent the index data from being sent to the peer cluster. The peer cluster generates index data by itself. This parameter is supported only in versions earlier than MRS 3.x.

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.

Related Interfaces

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. • To avoid wasting storage and processing resources, do not create two indexes on the same column that has different index names. • 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**, try again. • The index column family is selected from the 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 families exist in the
	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
			table in the sequence of the preceding priorities. If the column families do not exist, the system sets the first non-exist column family 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 index column family is **@** by default. – 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.

<table>
<tr><th>Operation</th><th>API</th><th>Description</th><th>Precautions</th></tr>
<tr><td rowspan="2">Delete an index.</td><td>dropIndices()</td><td>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.</td><td rowspan="2">● 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.</td></tr>
<tr><td>dropIndicesWithData()</td><td>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.</td></tr>
</table>

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 Using 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

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.

- Use the fully matched index.

  Scenario: There are two indexes available, one for **col1&col2** and the other for **col1**.

  In this case, 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, 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'}}

8.6.1.2 Loading HBase 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.

Prerequisites

- The client has been installed. For details, see **Installing a Client**.
- You have created a component service user with required permissions. A machine-machine user needs to download the **keytab** file and a human-machine user needs to change the password upon the first login.

Using HIndexImportTsv to Generate HBase Local Secondary Index Data in Batches

1. Log in to the node where the client is installed as the client installation user.
2. Run the following commands to configure environment variables and authenticate the user:

   cd *Client installation directory*

   source bigdata_env

   kinit *Component service user* (Skip this step for clusters with Kerberos authentication disabled.)
3. 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
4. Run the following command to create the **bulkTable** table:

   hbase shell

   create '*bulkTable***', {NAME => 'info',COMPRESSION => 'SNAPPY', DATA_BLOCK_ENCODING => 'FAST_DIFF'},{NAME=>'address'}**

Run the **!quit** command to exit **hbase shell**.

5. 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 the 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 structure is as follows:

     indexNameN=>familyN :[columnQualifierN-> columnQualifierDataType],[columnQualifierM-> columnQualifierDataType];familyM:[columnQualifierO-> columnQualifierDataType]# indexNameN=> familyM: [columnQualifierO-> columnQualifierDataType]

     ▪ Column qualifiers are separated by commas (,). For example:
       index1 => f1:[c1-> String], [c2-> String]
     ▪ Column families are separated by semicolons (;). For example:
       index1 => f1:[c1-> String], [c2-> String]; f2:[c3-> Long]
     ▪ Multiple indexes are separated by the number sign (#). For example:
       index1 => f1:[c1-> String], [c2-> String]; f2:[c3-> Long]#index2 => f2:[c3-> Long]
     ▪ The following data types are supported by columns:
       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
```

6. 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
```

8.6.1.3 Using TableIndexer to Generate a Local HBase Secondary Index

Scenarios

TableIndexer allows you to quickly index data in HBase. With this tool, you can 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 data will be generated, which is time-consuming. If you use **addIndices()** to create an index, index data corresponding to table data will not be generated. You can use the TableIndexer tool to create indexes.
- If the index data is inconsistent with the table 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 table.
- For tables that do not have indexes, this tool allows you to add and build indexes at the same time.

How to Use TableIndexer

- **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 parameters are as follows:

  – **tablename.to.index**: indicates the name of a table for which an index is created.
  – **indexspecs.to.add**: indicates the mappings between the index name and the column in the table.
  – (Optional) **scan.caching**: indicates the number of cached rows to be passed to the scanner during data table scans. The value contains an integer.

  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 the data type. The 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'

  – **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'

  – **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 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**.

8.6.1.4 Migrating HBase Index Data

Scenario

The indexes used in MRS 1.7 or later are incompatible with secondary indexes used by HBase in earlier MRS versions. Therefore, you need to perform the following operations to migrate index data from an earlier version (MRS 1.5 or earlier) to MRS 1.7 or later.

Prerequisites

1. During data migration, the cluster of the old version must be MRS 1.5 or earlier, and the cluster of the new version must be MRS 1.7 or later.
2. Before data migration, you must have old index data.
3. A cross-cluster mutual trust relationship must be configured and the inter-cluster replication function must be enabled for a security cluster. For a common cluster, only the inter-cluster replication function needs to be enabled. For details, see **Configuring Cross-Cluster Mutual Trust Relationships**and **Enabling Inter-Cluster Copy to Back Up Data**.

Procedure

Migrate user data from an old cluster to a new cluster. To migrate data, you need to manually synchronize data of the old and new clusters in a single table by export, distcp, and import.

For example, the current old cluster has a user table (**t1**, index name: **idx_t1**) and its corresponding index table (**t1_idx**). Perform the following operations to migrate data.

1. Export table data from the old cluster.
```
hbase org.apache.hadoop.hbase.mapreduce.Export -Dhbase.mapreduce.include.deleted.rows=true
<tableName> <path/for/data>
```
   - *<tableName>*: Indicates a table name, for example, **t1**.
   - *<path/for/data>*: Indicates the path for storing source data, for example, **/user/hbase/t1**.

   Example: **hbase org.apache.hadoop.hbase.mapreduce.Export -Dhbase.mapreduce.include.deleted.rows=true t1 /user/hbase/t1**
2. Copy the exported data to the new cluster as follows:
```
hadoop distcp <path/for/data> hdfs://ActiveNameNodeIP:9820/<path/for/newData>
```
   - *<path/for/data>*: Indicates the path for storing source data in the old cluster, for example, **/user/hbase/t1**.
   - *<path/for/newData>*: Indicates the path for storing source data in the new cluster, for example, **/user/hbase/t1**.

   ActiveNameNodeIP indicates the IP address of the active NameNode in the new cluster.

   Example: **hadoop distcp /user/hbase/t1 hdfs://192.168.40.2:9820/user/hbase/t1**

   NOTE

   - Manually copy the exported data to HDFS of the new cluster, for example, **/user/hbase/t1**.
3. Use the HBase table user of the new cluster to generate HFiles in the new cluster.
```
hbase org.apache.hadoop.hbase.mapreduce.Import -Dimport.bulk.output=<path/for/hfiles>
<tableName><path/for/newData>
```
   - *<path/for/hfiles>*: Indicates the path of the HFiles generated in the new cluster, for example, **/user/hbase/output_t1**.
   - *<tableName>*: Indicates a table name, for example, **t1**.
   - *<path/for/newData>*: Indicates the path for storing source data in the new cluster, for example, **/user/hbase/t1**.

   Example:

   hbase org.apache.hadoop.hbase.mapreduce.Import -Dimport.bulk.output=/user/hbase/output_t1 t1 /user/hbase/t1
4. Import the generated HFiles to the table in the new cluster.

   The command is as follows:
```
hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles <path/for/hfiles> <tableName>
```
   - *<path/for/hfiles>*: Indicates the path of the HFiles generated in the new cluster, for example, **/user/hbase/output_t1**.

– *<tableName>*: Indicates a table name, for example, **t1**.

Example:

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /user/hbase/output_t1 t1

NOTE

1. The preceding shows the process of migrating user data. You only need to perform the first three steps to migrate the index data of the old cluster and change the corresponding table name to an index table name (for example, **t1_idx**).
2. Skip **4** when migrating index data.

5. Import index data to a table in the new cluster.

   a. Add an index the same as that of the user table of the previous version to the user table of the new cluster (the user table cannot contain a column family named **d**).

      The command is as follows:

```
hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -
Dtablename.to.index=<tableName> -Dindexspecs.to.add=<indexspecs>
```

      - *-Dtablename.to.index=<tableName>*: Indicates a table name, for example, **-Dtablename.to.index=t1**.
      - **-Dindexspecs.to.add=<indexspecs>**: Indicates the mapping between an index name and a column, for example, **-Dindexspecs.to.add='idx_t1=>info:[name->String]'**.

      Example:

      hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=t1 -Dindexspecs.to.add='idx_t1=>info:[name->String]'

      NOTE

      If a column family named **d** exists in the user table, you must use the TableIndexer tool to build index data.

   b. Run the LoadIncrementalHFiles tool to load the index data of the old cluster to a table in the new cluster.

      The command is as follows:

```
hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles </path/for/hfiles>
<tableName>
```

      - *</path/for/hfiles>*: Indicates the path of index data on HDFS. The path is the index generation path specified in **-Dimport.bulk.output**, for example, **/user/hbase/output_t1_idx**.
      - *<tableName>*: Indicates a table name of the new cluster, for example, **t1**.

      Example:

      hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /user/hbase/output_t1_idx t1

8.6.2 Improving HBase BulkLoad Data Migration

8.6.2.1 Importing HBase Data in Batches Using BulkLoad

Scenario

Import data in batches to HBase and create indexes as you need by running commands.

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 a MapReduce task fails to be executed, rectify the fault by referring to **Why Physical Memory Overflow Occurs If a MapReduce Task Fails?**.
- The data sources supported by BulkLoad are text files with separators.
- You have installed the client. 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 you want to create a secondary index when importing data in batches, pay attention to the following:
  - 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 BulkLoad

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/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 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 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, customize and import the **configuration.xml** template file.

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 a secondary index is created during batch data import. The first letter of the index type must be capitalized, for example, **type="String"**. In the following segment, **length="30"** indicates the index column **H_ID**. The column value 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>

                <indices>
                    <index name="IDX1">
                        <index_column family="f1">
                            <qualifier type="String" length="30">H_ID</qualifier>
                        </index_column>
                    </index>
                </indices>

            </qualifiers>
            <badlines>SMS_ID &lt; 7000 &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 the table to be operated.
- *<inputdir>*: indicates the 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**.
- To use transparent encryption is for HBase, the HBase user who runs the bulkload command must be added to the **hadoop** user group of the cluster (If the cluster is not the first one installed on FusionInsight Manager, the user group is **c**<*Cluster ID*>**_hadoop**, for example, **c2_hadoop**), and have 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>*
- Creating 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

8.6.2.2 Updating HBase Data in Batches Using BulkLoad

Scenario

HBase BulkLoad updates data in batches based on the row key naming rule, row key scope, field name, and field value.

Updating HBase Data in Batches Using BulkLoad

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

- During batch updating, the field value of the row that meets the requirements will be updated.
- Batch updating cannot be performed on fields where indexes are created.
- If you do not set the output file of the execution result, the default value is **/tmp/updatedata/***table name*.

8.6.2.3 Deleting HBase Data in Batches Using BulkLoad

Scenario

BulkLoad deletes data in batches based on the row key naming rule, row key scope, field name, and field value.

Deleting HBase Data in Batches Using BulkLoad

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

- 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.
- If you do not set output data file **delete.hfile.output** of the execution result, the default value is **/tmp/deletedata/***table name*.

8.6.2.4 Counting Rows in an HBase Table Using BulkLoad

Scenario

HBase BulkLoad collects 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.

8.6.2.5 BulkLoad Configuration File

Configuring a Custom Combined Rowkey

You can customize combined rowkeys when using the BulkLoad tool to import HBase data in batches. The BulkLoad combines rowkeys by applying rules to process multiple column names and generate new rowkeys.

NOTE

The column name consists of letters, digits, and underscores (_) and cannot contain any special characters.

The combined rowkey in the **configuration.xml** file has three parts: for example, the three characters starting from the second character of **SMS_ID** and **SMS_NAME** and **SMS_SERAIL** in reverse order. Each part is connected by an underscore (_).

```
<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 8-5 Rowkey segment process functions

Syntax	Description	Example Value
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.	For example, **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 to reverse "ABC" to "CBA".

Syntax	Description	Example Value
substring(data,Length1,Length2), or substring(data,Length3)	Used to subtract a string.	For example, **substring(data,1,5)** or **substring(data,3)** is used to subtract [1,5) or [3,data.length) from the string.
to_number("data")	Used to convert a string into a numeric value. The Long type numeric value is supported.	For example, **to_number("123")** is used to convert "123" into 123. **data** must be a numeric value.

Configuring Custom Rowkeys

You can customize combined rowkeys when using the BulkLoad tool to import HBase data in batches. You can write RowKey implementation code to import data based on the combined RowKeys.

To configure a custom rowkey, perform the following steps:

Step 1 When you use 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],
```

Implement the method in the interface:

```
byte[] getRowkeyBytes(String[] colsValues, RegulationDomain regulation)
```

- **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 is installed, 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 a Custom Combined Field

You can customize combined fields 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.

In the following example, the combined field **H_COMBINE_1** consists of the **SMS_ADDRESS** and **SMS_SNAME** fields.

```
<!-- Define composite columns -->
            <composite family="f2">
                <!-- Define the class name of the concatenated field. The class must not exist in the application.-->;
                <qualifier class="com.huawei.H_COMBINE_1">H_COMBINE_1</qualifier>
                <columns>
                    <column>SMS_ADDRESS</column>
                    <column>SMS_NAME</column>
                </columns>
            </composite>
```

Specifying Field Data Types

HBase BulkLoad can read native data files, map the data file fields to HBase fields, 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.

The following shows the configuration of specifying the data type of a field. In this example, the data types of the **SMS_ID**, **SMS_NAME**, and **SMS_ADDRESS** columns are specified.

```
        <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

Supported data types are: short, int, long, float, double, boolean, and string.

Defining Inapplicable Data Rows

BulkLoad allows you to specify data rows that are not applicable. These rows are stored in a separate file, rather than in HBase.

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.

Define the applicable data rows. The following is a configuration example: **SMS_ID < 7000 && SMS_NAME = = 'HBase'**.


```
<!-- Define bad line filter rule -->
<badlines>SMS_ID &lt; 7000 &amp;&amp; SMS_NAME == 'HBase'</badlines>
```

Table 8-6 describes the operators and parameter data types in the **<badlines>** tag.

Table 8-6 Operators and corresponding data types

<table>
<tr><th>Operator Type</th><th>Data Type</th></tr>
<tr><td>&&</td><td>Boolean</td></tr>
<tr><td>&</td><td>Integer</td></tr>
<tr><td>|</td><td>Integer</td></tr>
<tr><td>^</td><td>Integer</td></tr>
<tr><td>/</td><td>Digit</td></tr>
<tr><td>==</td><td>String</td></tr>
<tr><td>>=</td><td>Digit</td></tr>
<tr><td>></td><td>Digit</td></tr>
<tr><td><<</td><td>Integer</td></tr>
<tr><td><=</td><td>Digit</td></tr>
<tr><td><</td><td>Digit</td></tr>
<tr><td>%</td><td>Digit</td></tr>
<tr><td>*</td><td>Digit</td></tr>
<tr><td>!=</td><td>String</td></tr>
<tr><td>||</td><td>Boolean</td></tr>
<tr><td>+</td><td>Digit and string</td></tr>
<tr><td>>></td><td>Integer</td></tr>
<tr><td>-</td><td>String</td></tr>
<tr><td>>>></td><td>Integer</td></tr>
</table>

8.6.3 Configuring RSGroup to Manage RegionServer Resources

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.

NOTE

This section applies only to MRS 3.1.2 or later.

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, ensure that there are more redundant RegionServer nodes than the required in in the RSGroup. The required number of nodes is calculated by: 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 web UI 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 RSGroup are allocated to each AZ. This constraint does not apply to 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

8.7 HBase Performance Tuning

8.7.1 Improving HBase BulkLoad Performance

Scenario

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.

NOTE

This section applies to MRS 3.*x* and later versions.

Prerequisites

When using BulkLoad, the output path of the file has been specified using the **Dimporttsv.bulk.output** parameter.

Procedure

Add the following parameter to the BulkLoad command when performing a batch loading task:

Table 8-7 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

8.7.2 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.

NOTE

This section applies to MRS 3.*x* and later versions.

Procedure

Navigation path for setting parameters:

On FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services** > **HBase** > **Configurations** > **All Configurations**. Enter the parameter name in the search box, and change the value.

Table 8-8 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

8.7.3 Optimizing 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.

NOTE

This section applies to MRS 3.*x* and later versions.

Procedure

Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HBase** > **Configurations**, and configure the following parameters to optimize 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 8-10 and **hbase.regionserver.global.memstore.size** in Table 8-9.
  - 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 8-9 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 8-10 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 8-11 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

8.7.4 Improving HBase Real-time Write Performance

Scenario

Scenarios where data needs to be written to HBase in real time, or large-scale and consecutive put scenarios

NOTE

This section applies to MRS 3.*x* and later versions.

Prerequisites

The HBase put or delete interface can be used to save data to HBase.

Procedure

- **Data writing server tuning**

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HBase**, click **Configurations**, and click **All Configurations**. For details about how to configure, see **Modifying Cluster Service Configuration Parameters**.

Table 8-12 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=0x7FFFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFFE -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. Performing 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 a major compaction must be performed to reclaim more space, increase the parameter value to increase the execution period and reduce frequent resource occupation, in milliseconds.	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**

  Using Put List greatly improves data 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**

Table 8-13 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

Parameter	Description	Default Value
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

8.7.5 Improving HBase Real-Time Read Efficiency

Scenario

HBase data needs to be read.

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**

  Log in to FusionInsight Manager and choose **Cluster** > **Services** > **HBase**, click **Configurations**, and click **All Configurations**. For details about how to configure, see **Modifying Cluster Service Configuration Parameters**.

Table 8-14 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.	For versions earlier than MRS 3.x: ● HMaster: -server -Xms2G -Xmx2G -XX:NewSize=256M -XX:MaxNewSize=256M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:MaxDirectMemorySize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps

Parameter	Description	Default Value
		-XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M ● RegionServer: -server -Xms4G -Xmx4G -XX:NewSize=512M -XX:MaxNewSize=512M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:MaxDirectMemorySize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcIn

Parameter	Description	Default Value
		terval=0x7FFFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M For MRS 3.*x* or later: ● 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 -

Parameter	Description	Default Value
		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 ● RegionServer -server -Xms6G -Xmx6G -XX:NewSize=1024M -XX:MaxNewSize=1024M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSIniti

Parameter	Description	Default Value
		atingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -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

Parameter	Description	Default Value
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**

Table 8-15 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

Parameter	Description	Default Value
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

8.7.6 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 8-16**.

Table 8-16 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

Variable	Affected Role
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"
```

8.8 HBase O&M Management

8.8.1 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 8-17 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.

Type	Name	Description
	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 8-18 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 8-18 Log levels

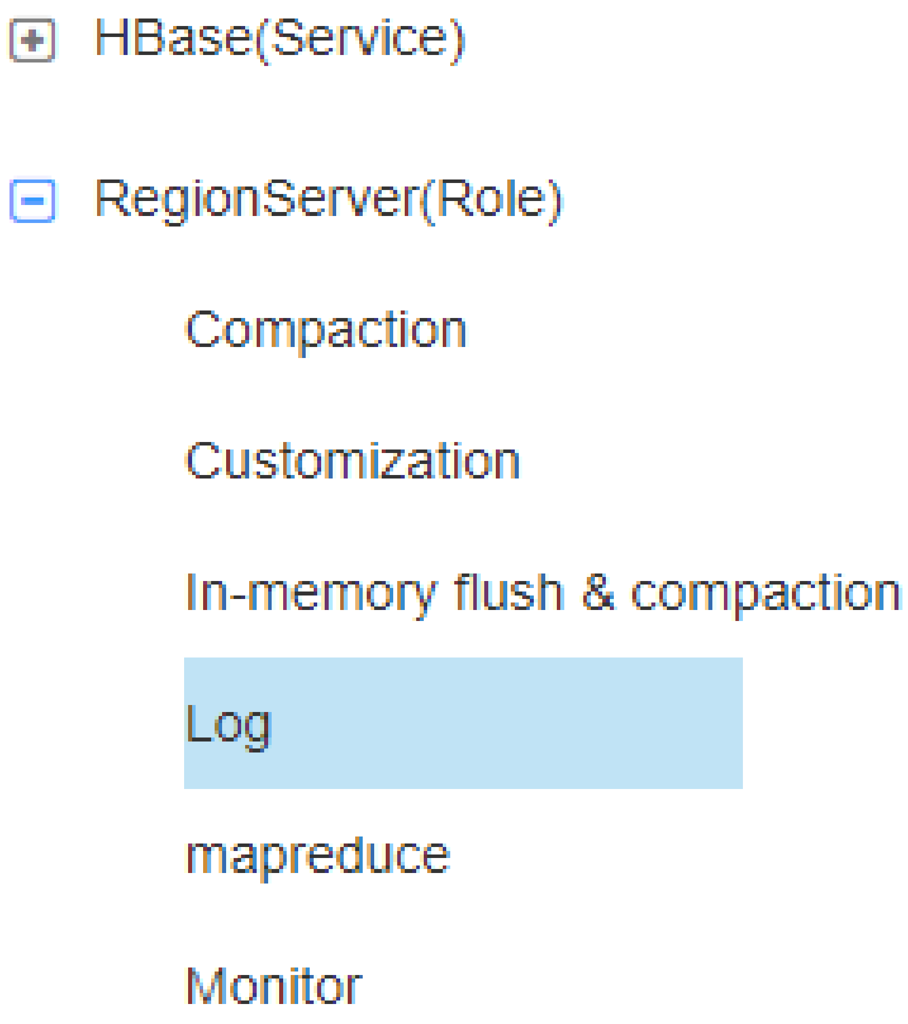
Level	Description
FATAL	Logs of this level record fatal error information about the current event processing that may result in a system crash.
ERROR	Logs of this level record error information about the current event processing, which indicates that system running is abnormal.

Level	Description
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 8-19 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)

Type	Component	Format	Example
	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)
	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)

8.8.2 HBase Common Configuration Parameters

NOTE

The operations described in this section apply only to clusters of versions earlier than MRS 3.x.

If the default parameter settings of the MRS service cannot meet your requirements, you can modify the parameter settings as required.

Step 1 Go to the cluster details page and choose **Components**.

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 Choose **HBase** > **Service Configuration** and switch **Basic** to **All**. On the displayed HBase configuration page, modify parameter settings.

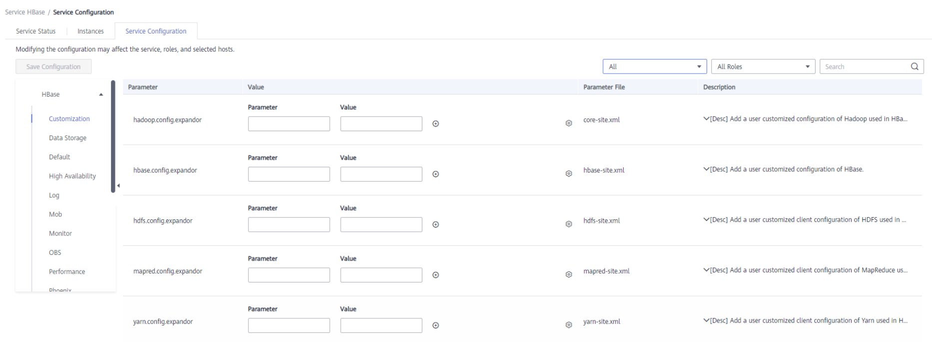


Table 8-20 HBase parameters

Parameter	Description	Value
hbase.regionserver.hfile.durable.sync	Whether to enable the HFile durability to make data persistence on disks. If this parameter is set to **true**, HBase performance is affected because each HFile is synchronized to disks by **hadoop fsync** when being written to HBase. This parameter exists only in MRS 1.9.2 or earlier.	Possible values are as follows: • **true** • **false** The default value is **true**.
hbase.regionserver.wal.durable.sync	Specifies whether to enable WAL file durability to make the WAL data persistence on disks. If this parameter is set to **true**, HBase performance is affected because each edited WAL file is synchronized to disks by **hadoop fsync** when being written to HBase. This parameter exists only in MRS 1.9.2 or earlier.	Possible values are as follows: • **true** • **false** The default value is **true**.

----End

8.8.3 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 Region Transition Recovery

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 lists the parameters for enabling this function.

Table 8-21 Parameters

Parameter	Description	Default Value
hbase.region.assignment.auto.recovery.enabled	Configuration parameter used to enable/disable the region assignment recovery thread feature.	true

8.8.4 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** to check the information.

For versions earlier than MRS 3.x, choose **Components** > **HDFS** > **Service Configuration** on the cluster details page. Switch **Basic** to **All**, and search for **hadoop.rpc.protection**.

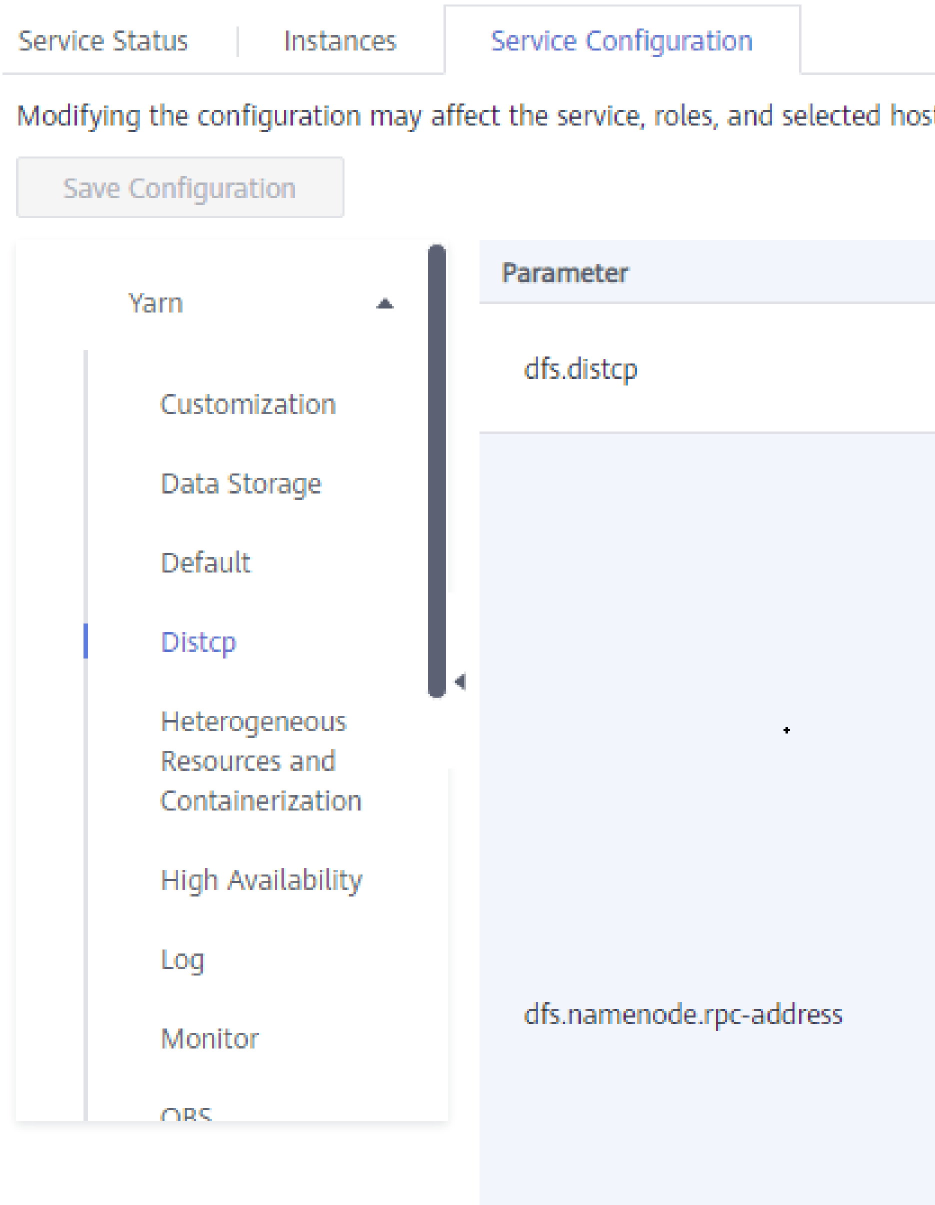
Procedure

Step 1 Go to the **All Configurations** page of the Yarn service. For details, see **Modifying Cluster Service Configuration Parameters**.

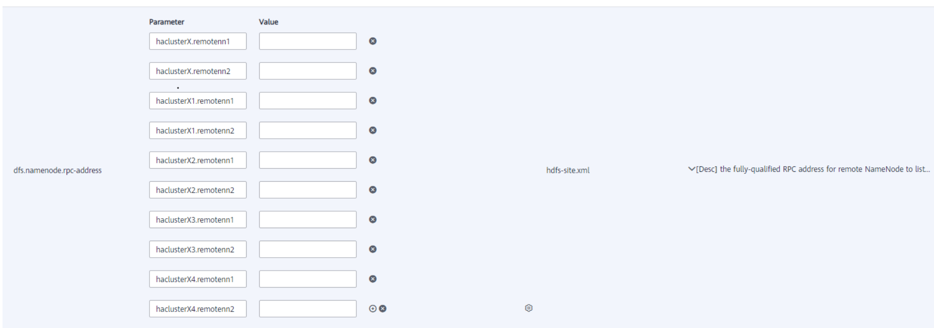
NOTE

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.



NOTE

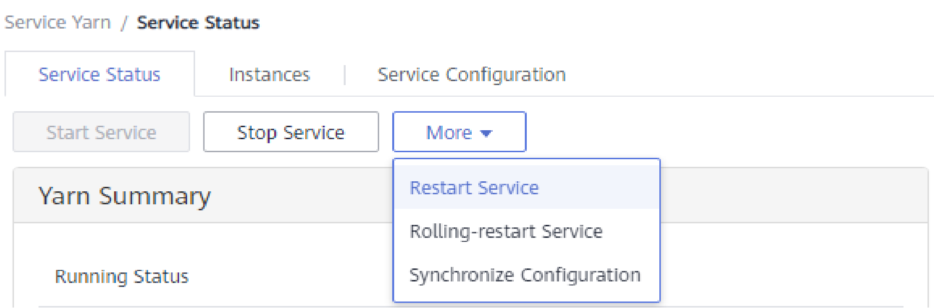
You can log in to FusionInsight Manager in MRS 3.x clusters, and choose **Cluster** > *Name of the desired cluster* > **Services** > **HDFS** > **Instance** to obtain the service IP address of the NameNode instance.

For versions earlier than MRS 3.x, on the cluster details page, choose **Components** > **HDFS** > **Instances** to obtain the service IP address of the NameNode instance.

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**. To obtain the NameNode RPC port, search for the **dfs.namenode.rpc.port** parameter on the HDFS service configuration page. The password 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

8.8.5 Configuring Automatic Data Backup for Active and Standby HBase Clusters

Prerequisites

1. Active and standby clusters have been installed and started.
2. Time is consistent between the active and standby clusters and the NTP service on the active and standby clusters uses the same time source.
3. When the HBase service of the active cluster is stopped, the ZooKeeper and HDFS services must be started and run.
4. ReplicationSyncUp must be run by the system user who starts the HBase process.
5. 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.

6. 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 **hbase shell** command on the client 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

1. 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.
2. 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.
3. 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.

4. 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.

8.8.6 Configuring HBase Cluster HA and DR

8.8.6.1 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.

NOTE

This section applies to MRS 3.*x* or later.

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

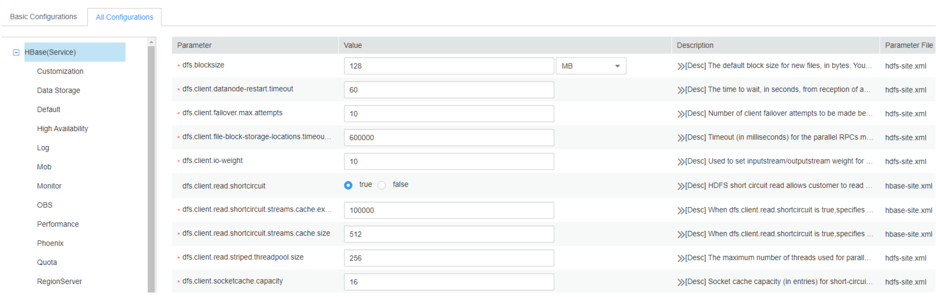
- Although DR provides the real-time data replication function, the data synchronization progress is affected by many factors, such as the service volume in the active cluster and the health status of the standby cluster. In normal cases, the standby cluster should not take over services. In extreme 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** > *Name of the desired cluster* > **Services** > **HBase** > **Configurations** and click **All Configurations**. The HBase configuration page is displayed.



Step 3 (Optional) **Table 8-22** describes the optional configuration items during HBase DR. You can set the parameters based on the description or use the default values.

Table 8-22 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.
	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** > *Name of the desired 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 installing the client

Step 8 Click **Save**. In the displayed dialog box, click **OK**. Restart the HBase service.

Step 9 In the active and standby clusters, choose **Cluster** > ***Name of the desired cluster*** > **Service** > **HBase** > **More** > **Download Client** to download the client and install it.

Adding the DR relationship between the active and standby clusters

Step 10 Log in as user **hbase** to the HBase shell page on the active cluster.

The initial password of user **hbase** is **Hbase@123**. For details, see **User Account List**.

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.
- Search for **hbase.master.kerberos.principal** and **hbase.regionserver.kerberos.principal** in the HBase **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.

- Create the **/hbase/replicationConf/*hbase.replication.cluster.id of the active cluster*** directory in the HDFS of the standby cluster.
- Copy the HBase client configuration file to the **/hbase/replicationConf/*hbase.replication.cluster.id of the active cluster*** directory of the HDFS of the standby cluster.

  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 timeColumn 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 8-23 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: • Create the **/hbase/replicationConf/*hbase.replication.cluster.id of the active cluster*** directory in the HDFS of the standby cluster. • 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.

Operation	Command	Description
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.
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.

8.8.6.2 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.

NOTE

This section applies to MRS 3.*x* or later.

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 **Switching Between Active and DR HBase Clusters**.

Disabling the write function of the active cluster

Step 2 Download and install the HBase client.

For details, see **Installing a Client (MRS 3.x or Later)**.

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:

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 preceding 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

8.8.6.3 Switching Between Active and DR HBase Clusters

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.

NOTE

This section applies to MRS 3.*x* or later.

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.

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

8.9 Common Issues About HBase

8.9.1 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.

8.9.2 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 on the HBase web UI 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*

8.9.3 What Should I Do If 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)

8.9.4 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\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 8-24 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**.

8.9.5 What Should I Do If Error Message "java.lang.UnsatisfiedLinkError: Permission denied" Is Displayed When I Start the HBase Shell?

Question

Why does the "java.lang.UnsatisfiedLinkError: Permission denied" exception occur when the HBase Shell is started?

Answer

During the execution of HBase Shell, JRuby creates a temporary file in **java.io.tmpdir**. The default path is **/tmp**. If the NOEXEC permission is set for the **/tmp** directory, the HBase Shell fails to be started and the error message "java.lang.UnsatisfiedLinkError: Permission denied" is displayed.

In this case, **java.io.tmpdir** must be set to a different path in **HBASE_OPTS/CLIENT_GC_OPTS**.

8.9.6 When Will the" Dead Region Servers" Information Displayed on the HMaster Web UI Be Cleared After a RegionServer Is Stopped?

Question

When will RegionServers in the Dead Region Servers state be cleared on the HMaster Web UI?

Answer

If a RegionServer stops suddenly, it enters the Dead Region Servers state on the HMaster web UI. When the stopped RegionServer restarts and reports a success message to HMaster, the "Dead Region Servers" information disappears from HMaster web UI.

If the HMaster active/standby switchover is successful, the "Dead Region Servers" information will also be deleted from the HMaster web UI.

If the active HMaster in some regions stops responding, the standby HMaster becomes active and the previous active HMaster becomes a dead RegionServer. If the HMaster active/standby switchover is successful, the "Dead Region Servers" information will also be deleted from the HMaster web UI.

8.9.7 What Can I Do If a Message Indicating Insufficient Permission Is Displayed When I Access HBase Phoenix?

Question

When a tenant accesses Phoenix, an error message is displayed indicating that the tenant has insufficient rights.

Answer

A tenant must be associated with the HBase service and YARN queue when being created.

To operate Phoenix, tenants need to obtain the RWX permission on Phoenix system catalogs.

The following is an example:

The created tenant is **hbase**. Log in to the HBase Shell as the **admin** user and run the **scan 'hbase:acl'** command to query the role of the tenant. The role is **hbase_1450761169920** (in tenant name_timestamp format).

Run the following commands to grant permissions (if the Phoenix system table is not generated, log in to the Phoenix client as user **admin** and then return to the HBase Shell for authorization):

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 to tenant **hbase**. Then user **phoenix** can access the Phoenix client.

8.9.8 How Do I Restore an HBase Region in Overlap State?

Question

When the hbck tool is used to check the region status in MRS 3.x and later versions, 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 overlapped table.

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 the restored table overlaps.

- 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 the restored table overlaps.

- If overlapping does not exist, no further action is required.
- If overlapping still exists, start from **Step 1** to perform the recovery again.

----End

8.9.9 Phoenix BulkLoad Use Restrictions

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 an index.

hbase org.apache.phoenix.mapreduce.index.IndexTool --data-table TEST_TABLE --index-table TEST_TABLE_INDEX --output-path /user/test_table

----End

8.9.10 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.

```
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(RangerAuthorizationCoprocessor.java:654)
at 
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.requirePermission(RangerAuthorizationCoprocessor.java:772)
at 
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.preCreateTable(RangerAuthorizationCoprocessor.java:943)
at 
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.preCreateTable(RangerAuthorizationCoprocessor.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

CTBase users can 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.

8.10 HBase Troubleshooting

8.10.1 The HBase Client Failed to Connect to the 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 8-25** describes them.

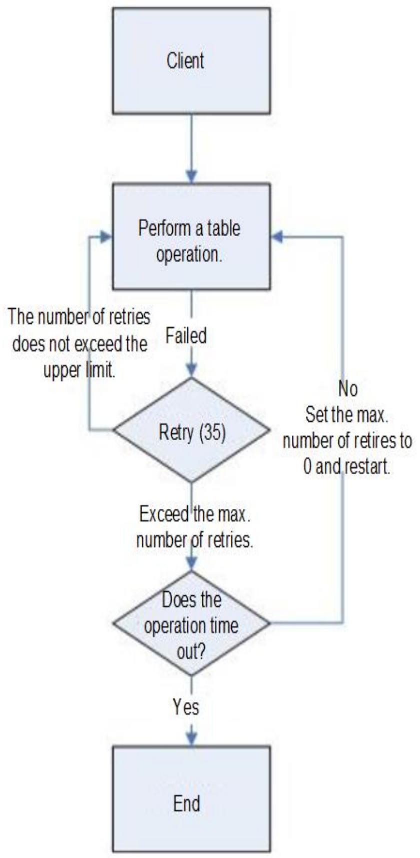
Set the following parameters in the ***Client installation path*/HBase/hbase/conf/hbase-site.xml** configuration file:

Table 8-25 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
hbase.client.retries.number	Maximum retry times supported by all retryable operations.	35

Figure 8-2 describes the working principles of retry and timeout.

Figure 8-2 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**.

8.10.2 An Exception Occurred When HBase Deletes and Creates a 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 what described in **1** happens, 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.

8.10.3 Other Services Are Unstable When Too Many HBase Connections Occupy the Network Ports

Question

What should I do if HBase connections occupy too many network ports and other services become unstable.

Answer

Run the OS command ***lsof*** or ***netstat***. A large number of TCP connections are in the **CLOSE_WAIT** state and the connection holder is HBase RegionServer. As a result, network ports may be used up or the number of HDFS connections may exceed the threshold, causing service instability. HBase connections can get stuck in a **CLOSE_WAIT** state due to the HBase mechanism.

HBase creates an HDFS connection when it creates or loads a StoreFile, which is HBase data stored in HFlile. When the creation or load task is complete, HDFS can close the connection. However, to ensure real-time response, HBase keeps the connection open to access the data file when needed. This results in the **CLOSE_WAIT** status, indicating that the client is waiting for a close request.

A StoreFile is created when HBase executes **flush**.

Data is written to HBase memory first, stored in the memstore. The ***flush*** operation is triggered when memory usage reaches the threshold or a manual flush command is executed, writing data to HDFS.

Solutions:

To reduce HBase port usage, control the number of StoreFiles by triggering HBase compaction, which combines files. Trigger HBase compaction by either of the following methods:

Method 1: Use the HBase shell client and perform the ***major_compact*** operation on the client.

Method 2: Write the HBase client code and call the **compact** method in the **HBaseAdmin** class to trigger the compaction in HBase.

If the compaction cannot solve HBase port occupation, the HBase usage has reached the limit. Check the following items:

- Whether the initial number of regions of the table is proper
- Whether useless data exists

If useless data exists, delete the data to reduce the number of files stored in HBase. If the problem persists, scale-out is required.

8.10.4 HBase BulkLoad Tasks of 210,000 Map Tasks and 10,000 Reduce Tasks failed To Be Executed

Question

The HBase bulkLoad task (a single table contains 26 TB data) has 210,000 maps and 10,000 reduce tasks (in MRS 3.x or later), 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 can 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.

8.10.5 Modified and Deleted Data Can Still Be Queried by the Scan Command

Question

Why can I still query the modified and deleted data by running the following **scan** command?

```
scan '<table_name>',{FILTER=>"SingleColumnValueFilter('<column_family>','column',=,'binary:<value>')"}
```

Answer

When you query a table in HBase, all versions of queried column values are searched by default, including deleted or modified values. If a row fails to be hit (that is, the column cannot be matched in the row), HBase queries the row.

If you only need to query the latest value of a table and the rows that are hit, run the following statement:

```
scan '<table_name>',
{FILTER=>"SingleColumnValueFilter('<column_family>','column',=,'binary:<value>',true,true)"}
```

This command filters out the rows that fail to be hit and queries the latest version of the table data. That is, the values before modification and deleted values are not queried.

NOTE

The 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**: indicates the column family of the column you want to query.
- **qualifier**: indicates the column you want to query.
- **compareOp**: indicates the comparison operator, such as **=** and **>**.
- **comparator**: indicates the target value to be searched for.
- **filterIfMissing**: indicates whether a row is filtered if the column cannot be matched in this row. The default value is **false**.
- **latestVersionOnly**: indicates whether only values of the latest version will be queried. The default value is **false**.

8.10.6 Failed to Create Tables When the Region is in FAILED_OPEN State

Question

What should I do if I fail to create tables when the region is in FAILED_OPEN state?

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:

To do so, perform the following steps:

1. Run the following command on the cluster client to repair the tables:
   hbase hbck -j ${CLIENT_HOME}**/HBase/hbase/tools/hbase-hbck2-1.1.0-h0.cbu.mrs.*.jar setTableState** <table_name> **ENABLED**
2. Go to the HBase Shell and run the following commands to clear the table:
   disable *'<table_name>'*
   drop *'<table_name>'*
3. Run the table creation command to create the table again.

8.10.7 How to Delete the residual Table Name on the ZooKeeper table-lock Node After a Table Creation Failure

Question

In security mode, a table fails to be created, and a new table name remains on the table-lock node (default path: **/hbase/table-lock**) of ZooKeeper. How do I clear the table name?

Answer

The detailed procedure is as follows:

1. In the environment where the client is installed, perform kinit authentication as user **hbase**.
2. Run the ***hbase zkcli*** command to go to the ZooKeeper CLI.
3. Run the ***ls /hbase/table*** command in the ZooKeeper command line to check whether the new table name exists.
   – If yes, run the ***ls /hbase/table-lock*** command to check whether the new table name exists. If the new table exists, run the **delete /hbase/table-lock/***<table>* command to delete the table. *<table>* indicates the residual table name.

– If no, no further operation is required.

8.10.8 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(DirectoryWit
hQuotaFeature.java:211)
?at
org.apache.hadoop.hdfs.server.namenode.DirectoryWithQuotaFeature.verifyQuota(DirectoryWithQuotaFeatu
re.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

8.10.9 HMaster Failed to Be Started After the OfflineMetaRepair Tool Is Used to Rebuild Metadata

Question

After the OfflineMetaRepair tool is used to rebuild metadata, the namespace table allocation times out during HMaster startup. Why does the startup fail?

HMaster outputs the following FATAL message to indicate that the operation is terminated:

```
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

During startup, HMaster waits for WAL splitting on all region servers to ensure metadata consistency before rebuilding metadata with OfflineMetaRepair. When WAL splitting is complete, HMaster allocates user regions. However, this process can be slow if the cluster is abnormal, due to factors like many WALs, slow I/Os, or unstable region servers.

To ensure successful splitting of all region server WALs on on HMaster, perform the following steps:

1. Ensure that the cluster is stable and no other problems exist. If any problem occurs, rectify it first.
2. Set the **hbase.master.initializationmonitor.timeout** parameter to a large value. The default value is **3600000** milliseconds.
3. Restart the HBase service.

8.10.10 FileNotFoundException Is Frequently Printed in 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 8-26** describes the parameter.

Table 8-26 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

8.10.11 Data Is Successfully Imported Using HBase BulkLoad, but Different Results May Be Returned To the Same Query

Question

When importing data using HBase bulkload, data with identical rowkey values can be successfully imported, but queries may yield different results.

Answer

Data with the same rowkey values is loaded to HBase in sequence. HBase takes the data with the latest timestamp as the latest data. In default queries, only the latest data is returned for data with the same rowkey value, unless a specific timestamp is specified.

When you use bulkload to load data, HFiles are generated quickly in memory. However, this can cause query results to be disordered if data has the same rowkey value and timestamp.

When creating tables and loading data, ensure that rowkey values are designed to prevent duplicate rowkey values within the same data file.

8.10.12 HBase Data Restoration Task Failed to Be Rolled Back

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

8.10.13 RegionServer Failed to Be Started When GC Parameters Xms and Xmx of HBase RegionServer Are Set to 31 GB

Question

(MRS 3.x and later versions) 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

8.10.14 When LoadIncrementalHFiles Is Used to Import Data in Batches on Cluster Nodes, the Insufficient Permission Error Is Reported

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

8.10.15 "import argparse" Is Reported 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 Using HDFS

9.1 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 9-1 HDFS directory structure (applicable to versions earlier than MRS 3.x)

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/tmp/spark/sparkhive-scratch	Fixed directory	Stores temporary files of metastore sessions in Spark JDBCServer.	No	Failed to run the task.
/tmp/sparkhive-scratch	Fixed directory	Stores temporary files of metastore session that are executed using Spark CLI.	No	Failed to run the task.
/tmp/carbon/	Fixed directory	Stores the abnormal data in this directory if abnormal CarbonData data exists during data import.	Yes	Error data is lost.

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/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/logs	Fixed directory	Stores the collected MR task logs.	Yes	MR task logs are lost.
/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.
/tmp/hive	Fixed directory	Stores Hive temporary files.	No	Failed to run the Hive task.
/tmp/hive-scratch	Fixed directory	Stores temporary data (such as session information) generated during Hive running.	No	Failed to run the current task.

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/user/{user}/.sparkStaging	Fixed directory	Stores temporary files of the SparkJDBCServer application.	No	Failed to start the executor.
/user/spark/jars	Fixed directory	Stores running dependency packages of the Spark 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/mapred	Fixed directory	Stores Hadoop-related files.	No	Failed to start Yarn.
/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.
/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.

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/sparkJobHistory	Fixed directory	Stores Spark event log 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.
/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.

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/ats/active	Fixed directory	HDFS path used to store the timeline data of running applications.	No	Failed to run the **tez** task after the directory deletion.
/ats/done	Fixed directory	HDFS path used to store the timeline data of completed applications.	No	Automatically created after the deletion.
/flink	Fixed directory	Stores the checkpoint task data.	No	Failed to run tasks after the deletion.

Table 9-2 Directory structure of the HDFS file system (applicable to MRS 3.*x* or later)

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.
/tmp/carbon/	Fixed directory	Stores the abnormal data in this directory if abnormal CarbonData data exists during data import.	Yes	Error data is lost.

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/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.
/tmp/hive-scratch	Fixed directory	Stores temporary data (such as session information) generated during Hive running.	No	Failed to run the current task.

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/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-*3.1.1*.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.
/user/omm-bulkload	Temporary directory	Stores HBase batch import tools temporarily.	No	Failed to import HBase tasks in batches.

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/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.
/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.

Path	Type	Function	Whether the Directory Can Be Deleted	Deletion Consequence
/hbaseFileStream	Fixed directory	Stores HFS files.	No	The HFS file is lost and cannot be restored.

9.2 HDFS User Permission Management

9.2.1 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

- This section applies to MRS 3.*x* or later.
- 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 9-3**.

Table 9-3 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**.

Step 4 Click **OK**. Return to the **Role** page.

----End

9.2.2 Granting HDFS Users the Permission to Access HDFS Files

Configuring HDFS Directory Permission

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.

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

Setting Permissions on HDFS Files and Directories

HDFS allows users to modify the default permissions of files and directories. The default mask provided by the HDFS for creating file and directory permissions is **022**. If you have special requirements for the default permissions, you can set configuration items to change the default permissions.

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 9-4 Parameters

Parameter	Description	Default Value
fs.permissions.umask-mode	This **umask** value (user mask) is used when the user creates files and directories in the HDFS on the clients. This parameter is similar to the file permission mask on Linux. The parameter value can be in octal or in symbolic, for example, **022** (octal, same as **u=rwx,g=r-x,o=r-x** in symbolic), or **u=rwx,g=rwx,o=** (symbolic, same as **007** in octal). **NOTE** The octal mask is opposite to the actual permission value. You are advised to use the symbol notation to make the description clearer.	022

9.3 Using the HDFS Client

Scenario

This section describes how to use the HDFS client in an O&M scenario or service scenario.

Prerequisites

- You have installed the client.

  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 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 needs to change the password upon the first login. (This operation is not required in normal mode.)

Using the HDFS 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 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 5 Run the HDFS Shell command. Example:

hdfs dfs -ls /

----End

Common HDFS Client Commands

The following table lists common HDFS client commands.

Table 9-5 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

- Question 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?

  Answer: 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**

- Question 2: How do I set the log level when the HDFS client is running?

  Answer: By default, the logs generated during the running of the HDFS client are printed to the management 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

- Question 3: How do I permanently delete HDFS files?

  Answer: 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*.

9.4 Using Hadoop

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.

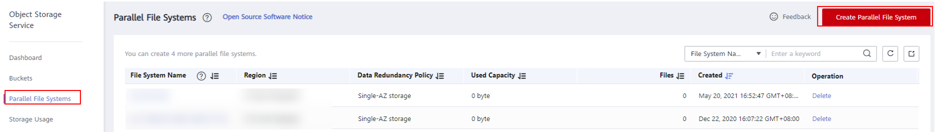
There is no format requirement for data files. Prepare one or more **.txt** files. The following are examples of the **.txt** file:

```
qwsdfhoedfrffrofhuncckgktpmhutopmma
jjpsffjfjorgjgtyiuyjmhombmbogohoyhm
jhheyeombdhuaqqiquyebchdhmamdhdemmj
doeyhjwedcrfvtgbmojiyhhqssddddddfkf
kjhhjkehdeiyrudjhfhfhffooqweopuyyyy
```

Step 3 Upload data to OBS.

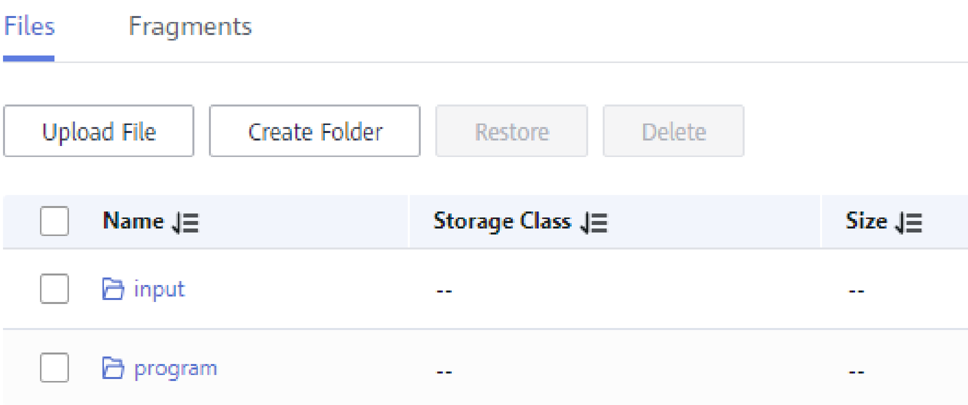
1. Log in to OBS Console.
2. Click **Parallel File System** and choose **Create Parallel File System** to create a file system named **wordcount01**.

   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 9-1**.

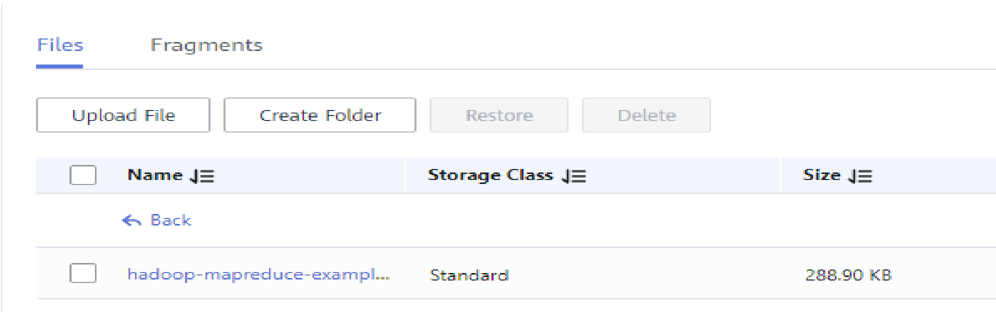
   Figure 9-1 Folder list of the wordcount01 file system

   – **program**: stores user programs.
   – **input**: stores user data files.

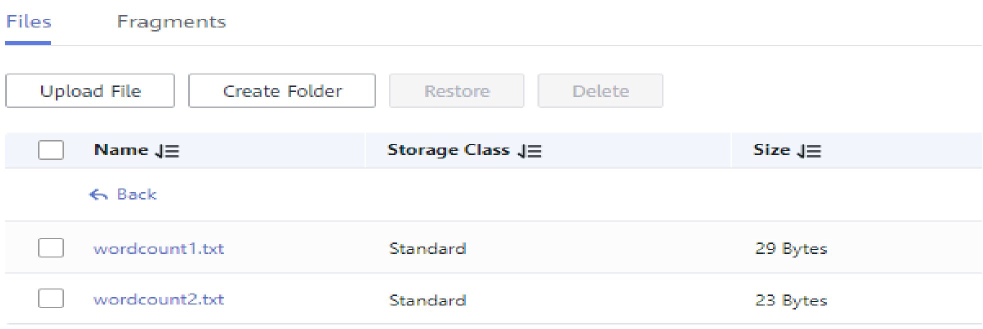
4. Go to the **program** folder, choose **Upload File** > **add file**, select the program package downloaded in **Step 1** from the local host, and click **Upload**. After the upload is complete, the page shown in **Figure 9-2** is displayed.

   Figure 9-2 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 9-3** is displayed.

Figure 9-3 Data file list



Step 4 Log in to the MRS console. In the navigation pane on the left, choose **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:

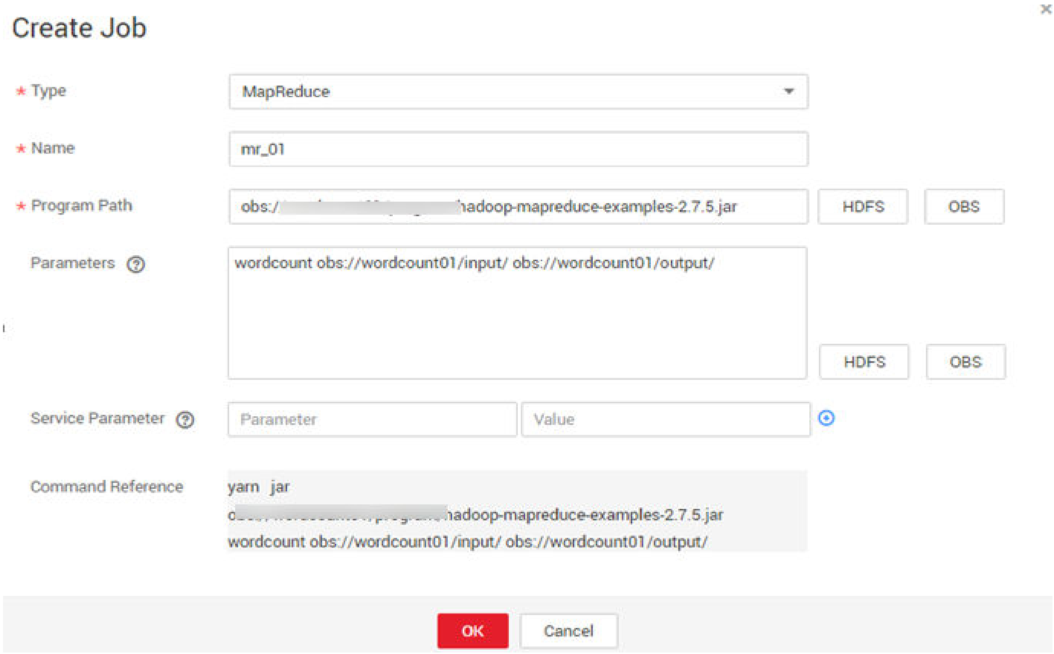
1. In the **Dashboard** tab of the cluster, check whether the agency parameter has a value and whether the bound agency has the permission to operate an OBS file system.

   – 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.

Figure 9-4 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.

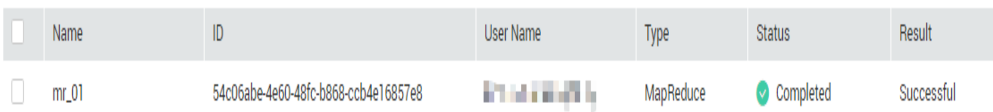
After a job is submitted successfully, it is in the **Accepted** state by default. You do not need to manually execute the job.

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 9-5**.

Figure 9-5 Job list



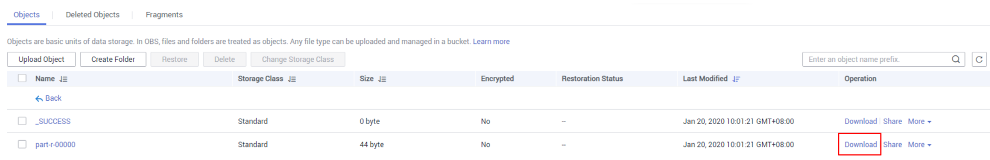
Name	ID	User Name	Type	Status	Result
mr_01	54c06abe-4e60-48fc-b868-ccb4e16857e8	[illegible]	MapReduce	Completed	Successful

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 9-6**.

Figure 9-6 Output file list



----End

9.5 Configuring the Recycle Bin Mechanism

Scenario

When a file is deleted in HDFS, it is moved to the recycle bin, or trash, instead of being immediately cleared. This allows for the recovery of deleted data in case of accidental deletions. After the aging time expires, a deleted file becomes an aging file and is cleared by the system or manually by users.

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

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 9-6 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

Parameter	Description	Default Value
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

9.6 Configuring HDFS DataNode Data Balancing

Scenario

NOTE

This section applies to MRS 3.*x* or later.

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 9-7** and **Figure 9-8**, respectively.

Figure 9-7 DataNode disk usage before balancing

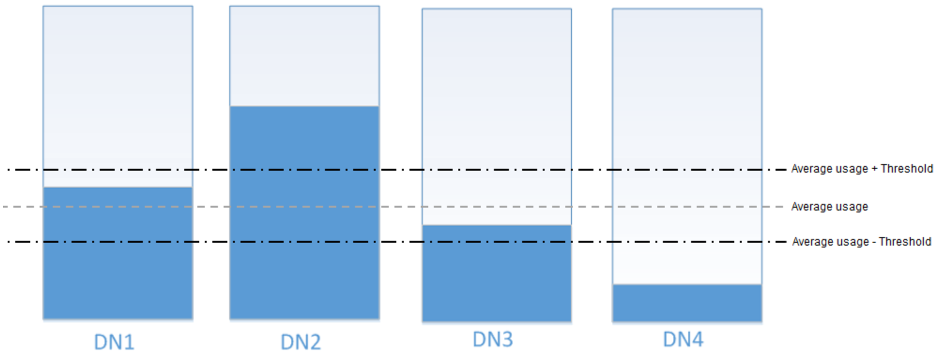
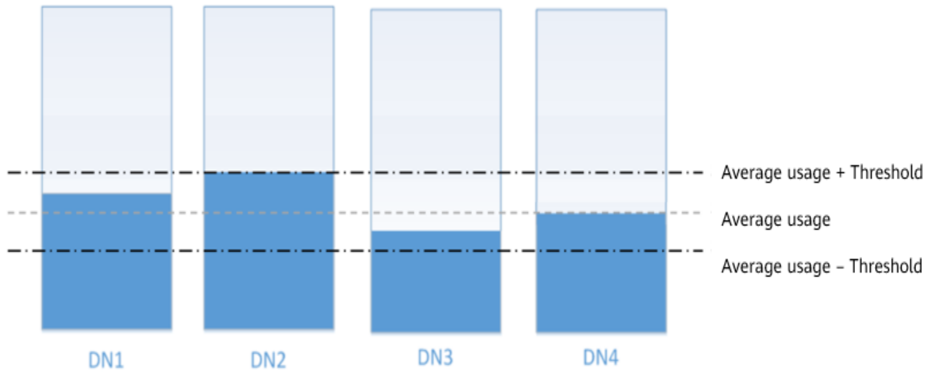


Figure 9-8 DataNode disk usage after balancing



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

You have installed the HDFS client.

Configuring a Balancing Task

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 limit, 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

- /opt/client indicates the client installation directory. Replace it with the directory you use.
- 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.
- 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

Setting Automatic Execution of the Balancing Task

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > **Services** > **HDFS**. Click **Configurations** then **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 9-7 describes the expression for modifying this parameter. * indicates consecutive time segments.

Table 9-7 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 9-7 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 9-8**.

Table 9-8 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

Parameter	Parameter description	Default Value
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

9.7 Configuring HDFS Disk Balancing

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.

NOTE

This section applies to MRS 3.*x* or later.

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 9-9 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 9-10**.	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
dfs.disk.balancer.block.tolerance.percent	Specifies the difference threshold between the data storage capacity and optimal 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

Table 9-10 lists the CRON expressions used for HDFS disk balancing. To use this function, set **dfs.disk.balancer.auto.enabled** to **true**. Set other parameters based on the cluster status.

Table 9-10 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 9-11 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.

Syntax	Description
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

- The **supergroup** permission is required for running this command on the client. You can run the command as the system user **hdfs** who has permission to use 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 9-11**. For more parameters to be configured for each command, run the **hdfs diskbalancer -help** *<command>* command to view detailed information.
- To troubleshoot performance issues, check the cluster event information for HDFS disk balancing events. If such events occurred, 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.

9.8 Using HDFS Mover to Migrate Data

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.

NOTE

This section applies to MRS 3.*x* or later.

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 9-12 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 9-13**.	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 9-13 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.

How to Use

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

The **supergroup** permission is required for running this command on the client. You can run the command as the system user **hdfs** who has permission to use the HDFS service. Alternatively, you can create a user with the **supergroup** permission in the cluster and then run the command.

9.9 Configuring the Label Policy (NodeLabel) for HDFS File Directories

Scenario

You need to configure the nodes for storing HDFS file data blocks based on data features. You can specify which DataNodes store data blocks for files by setting a label expression to match an HDFS directory/file and assigning one or more labels to each DataNode.

To place blocks with a label-based strategy, DataNodes that match the file's label expression are selected and then a suitable node among them is chosen.

NOTE

This section applies to MRS 3.*x* or later.

- 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 9-9 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 9-9**, 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, there can be multiple DataNodes of these racks under each label, to ensure reliability of data block placement.

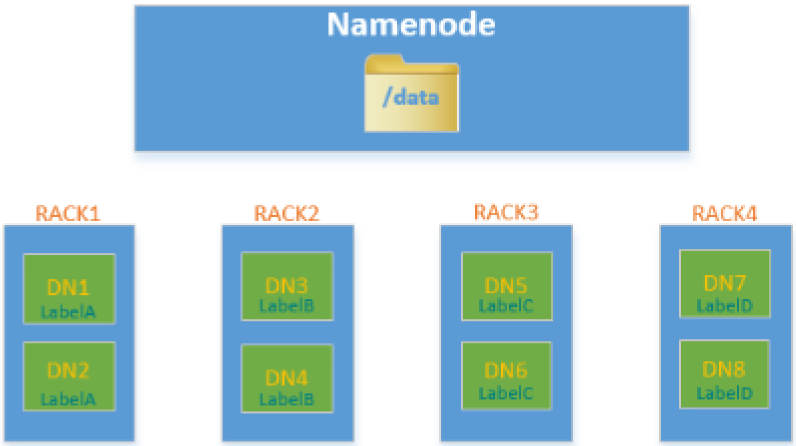
- Scenario 2: Specifying replica location when there are multiple racks

  Scenario description:

  In a heterogeneous cluster, allocate nodes with high reliability for storing important business data. Specify the replica locations using label expressions, and store one replica of file data blocks on a high-reliability 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 9-10 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 9-14 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
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 ranging 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, run the **hdfs nodelabel -listNodeLabels** command to view the label information of each DataNode.

Block Replica Location Selection

NodeLabel supports different label 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 copies exceeds the value of **dfs.replication**, HDFS deletes redundant block copies to improve resource utilization. **dfs.replication** indicates the number of file copies allowed. Go to the HDFS service configuration page by referring to **Modifying Cluster Service Configuration Parameters**, and search for the parameter to view its value.

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.

9.10 Configuring NameNode Memory Parameters

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 9-15 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

Parameter	Description	Default Value
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=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -Djdk.tls.ephemeralDHKeySize=2048

9.11 Setting the Number Limit of HBase and HDFS Handles

Scenario

When an HDFS file is opened and the number of handles is limited, the following error occurs:

```
IOException (Too many open files)
```

Set the number of HBase and HDFS handles by referring the following content.

Setting the Number Limit of HBase and HDFS Handles

To increase the number of handles for each user, contact the cluster administrator. 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 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

9.12 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 9-16 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

9.13 Enterprise-Class Enhancements of HDFS

9.13.1 Configuring Replica Replacement Policy for DataNodes with Inconsistent Capacity

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:

- First replica: stored on the node where the client resides.
- Second replica: stored on DataNodes of the remote rack.
- 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:

- First replica: stored on the DataNode where the client resides (the same as the default storage policy).
- 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.
- 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**.

Table 9-17 Parameters

Parameter	Description
dfs.block.replicator.classname	Specifies the DataNode replica placement policy. The default value is **org.apache.hadoop.hdfs.server.blockmanagement.AvailableSpaceBlockPlacement-Policy**.

Step 3 Save the modified configuration. Restart the expired service or instance for the configuration to take effect.

----End

9.13.2 Configuring Reserved Percentage of Disk Usage on DataNodes

Scenario

When the YARN local directory and DataNode directory are on the same disk, the larger disk can handle more tasks, storing more intermediate data in the YARN local directory.

Currently, you can set **dfs.datanode.du.reserved** to configure 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 significant amount 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 9-18 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

9.13.3 Enabling the NameNode Blacklist

Scenario

NOTE

This section applies to MRS 3.*x* or later.

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

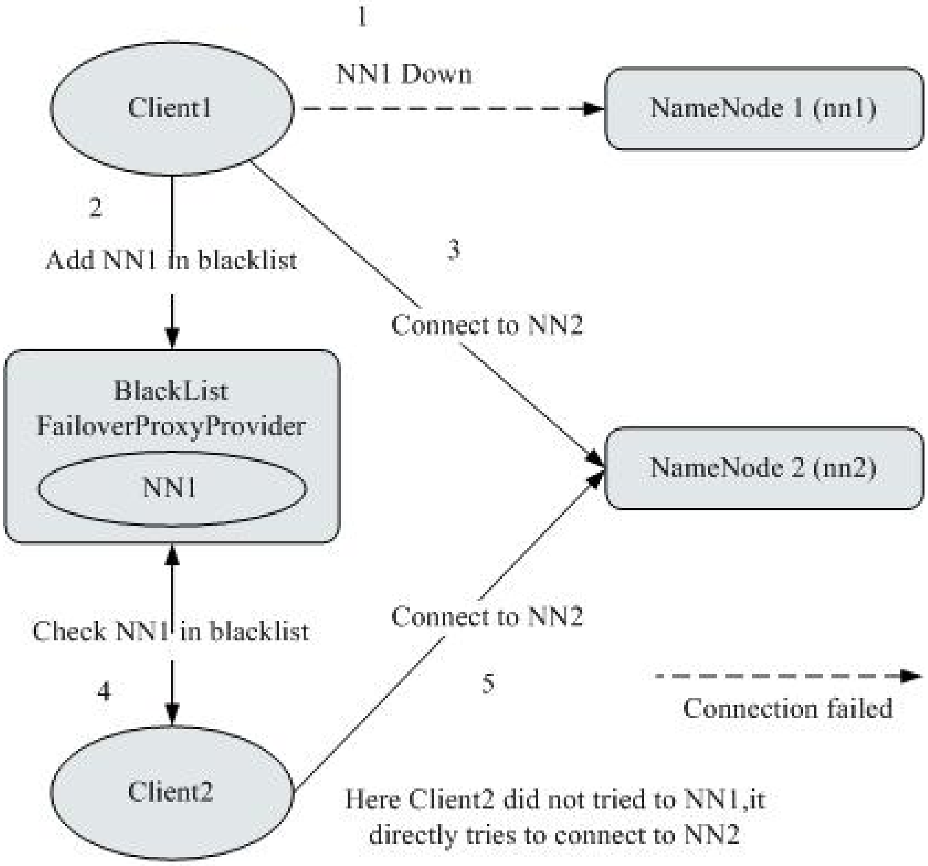
In the preceding cluster, if the active **nn1** cannot be accessed, client1 will retry the connection for 20 times. Then, a failover occurs, and client1 will connect to **nn2**. In the same way, other clients also connect to **nn2** when the failover occurs after retrying the connection to **nn1** for 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 9-11 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 9-19 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

9.13.4 Configuring Hadoop Data Encryption During Transmission

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:

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 9-20 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

9.14 HDFS Performance Tuning

9.14.1 Improving HDFS Write Performance

Scenario

Improve the HDFS write performance by modifying the HDFS attributes.

NOTE

This section applies to MRS 3.*x* or later.

Procedure

Navigation path for setting parameters:

On FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** and click **Configurations** then **All Configurations**. Enter a parameter name in the search box.

Table 9-21 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

9.14.2 Improving Read Performance By HDFS Client Metadata Caching

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.

This section applies to MRS 3.*x* or later.

Procedure

Navigation path for setting parameters:

On FusionInsight Manager, choose **Cluster** > **Services** > **HDFS**, click **Configurations** then **All Configurations**, and enter the parameter name in the search box.

Table 9-22 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();
```


9.14.3 Improving the HDFS Client Connection Performance with Active NameNode Caching

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.

NOTE

This section applies to MRS 3.*x* or later.

Procedure

Navigation path for setting parameters:

On FusionInsight Manager, choose **Cluster** > **Services** > **HDFS**, click **Configurations** then **All Configurations**, and enter the parameter name in the search box.

Table 9-23 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.

9.14.4 Optimization for Unstable HDFS Network

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 9-24 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

9.14.5 Optimizing HDFS NameNode RPC QoS

Scenarios

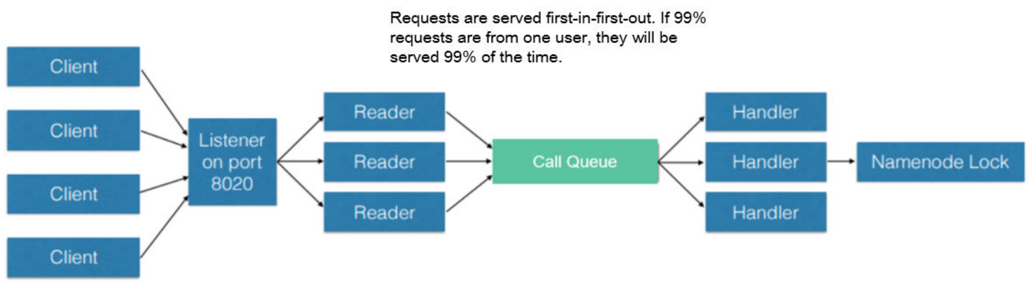
NOTE

This section applies to MRS 3.*x* or later.

Several finished Hadoop clusters are faulty because the NameNode is overloaded and unresponsive.

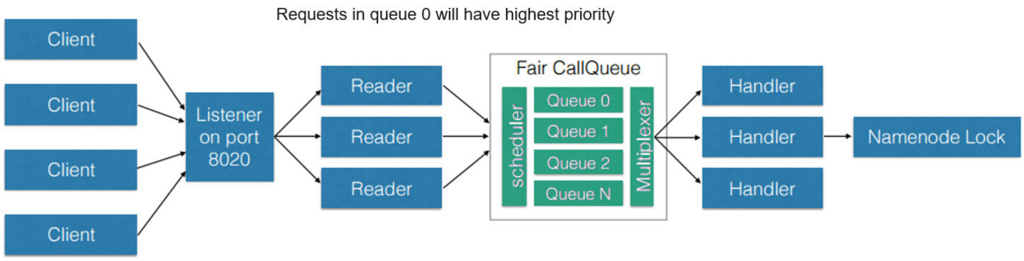
Such problem is caused by the initial design of Hadoop: In Hadoop, the NameNode functions as an independent part and in its namespace coordinates various HDFS operations, including obtaining the data block location, listing directories, and creating files. The NameNode receives HDFS operations, regards 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 9-12 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 9-13 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 9-25 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 9-26 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**.

9.14.6 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.

NOTE

This section applies to MRS 3.*x* or later.

Configuration

The **dfs.pipeline.ecn** parameter is available in MRS. 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 9-27 NameNode ECN configuration

Parameter	Description	Default Value
dfs.pipeline.ecn	After configuration, the DataNode can send blocking notifications to the client.	false

9.14.7 Performing Concurrent Operations on HDFS Files

Scenario

Performing this operation can concurrently modify file and directory permissions and access control tools in a cluster.

NOTE

This section applies to MRS 3.*x* or later.

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 a client that contains the HDFS client 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 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"

Run the following command to apply the modifications.

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.
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.

Command	Description	Function
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

9.14.8 Using the LZC Compression Algorithm to Store HDFS Files

Scenario

File compression can reduce the space occupied by stored files, and accelerate 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/.

NOTE

This section applies to MRS 3.*x* or later.

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 9-28 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.huawei.hadoop.datasight.io.compress.lzc.ZCodec

NOTE

- LZC does not support FSImage and SequenceFile compression.
- 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
- 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.
- 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.

9.15 HDFS O&M Management

9.15.1 HDFS Common Configuration Parameters

Page Access

Go to the HDFS configurations page by referring to **Modifying Cluster Service Configuration Parameters**.

Parameter Description

Table 9-29 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

9.15.2 HDFS Log Overview

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 9-30 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.
	<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.

Type	Name	Description
	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	Log that records HDFS service startup details
	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.
	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.

Type	Name	Description
	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 mount table management operations
	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 logs
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 9-31 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 9-31 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.

Level	Description
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 9-32 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>*	20xx-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*>	20xx-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)

9.15.3 Viewing the 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

To configure the following parameters, go to the HDFS service configuration page, refer to **Modifying Cluster Service Configuration Parameters**, and search for the corresponding parameters.

- 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 9-33 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 9-34 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 9-35 Number of DataNodes

Number of HDFS Blocks	Minimum Number of DataNode Roles
10,000,000	10,000,000 *3/5,000,000 = 6

Number of HDFS Blocks	Minimum Number of DataNode Roles
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

To configure the following parameters, go to the HDFS service configuration page, refer to **Modifying Cluster Service Configuration Parameters**, and search for the corresponding 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=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -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 9-36 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=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -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 9-37 DataNode JVM configuration

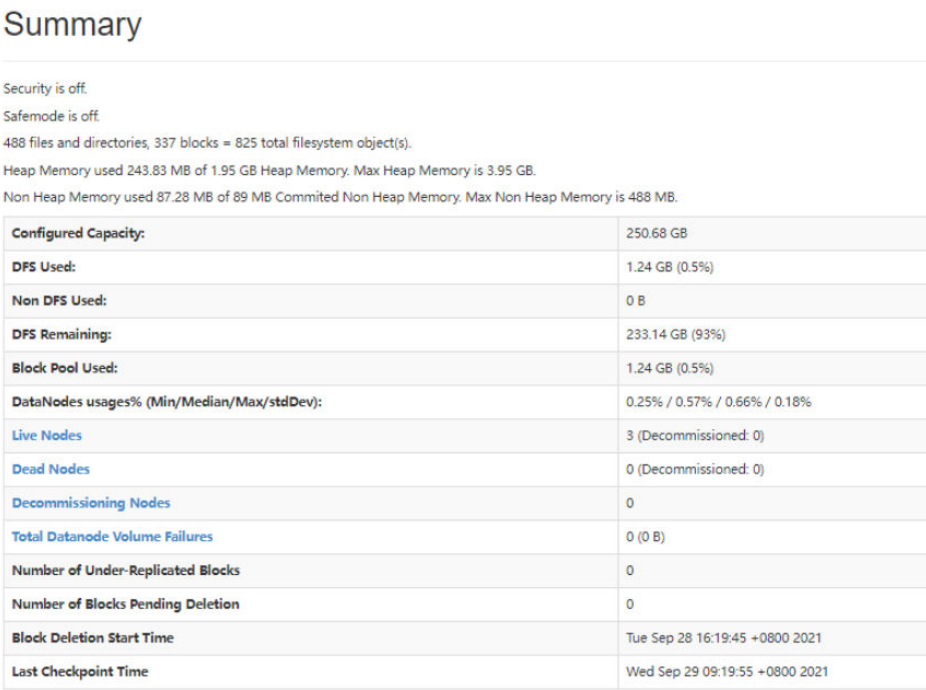
Average Number of Blocks in a DataNode Instance	Reference Value
2,000,000	-Xms6G -Xmx6G -XX:NewSize=512M -XX:MaxNewSize=512M
5,000,000	-Xms12G -Xmx12G -XX:NewSize=1G -XX:MaxNewSize=1G

  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

  For versions earlier than MRS 3.*x*: Log in to the MRS console, and choose **Components** > **HDFS** > **NameNode (Active)**. Click **Overview** and check the number of file objects, files, directories, or blocks in the HDFS in **Summary**.



Summary

Security is off.

Safemode is off.

488 files and directories, 337 blocks = 825 total filesystem object(s).

Heap Memory used 243.83 MB of 1.95 GB Heap Memory. Max Heap Memory is 3.95 GB.

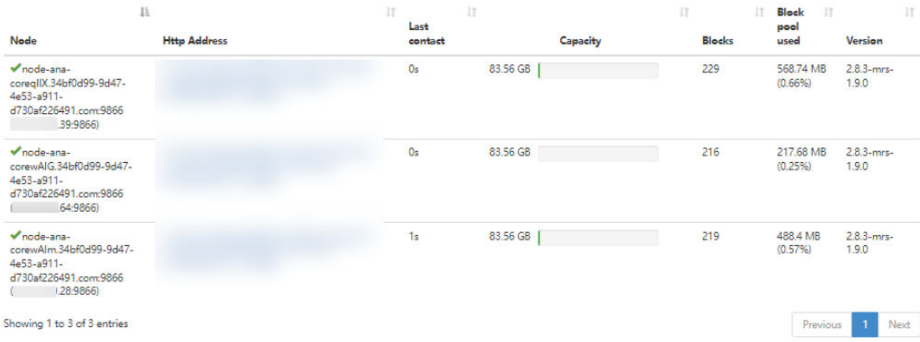
Non Heap Memory used 87.28 MB of 89 MB Commited Non Heap Memory. Max Non Heap Memory is 488 MB.

Configured Capacity:	250.68 GB
DFS Used:	1.24 GB (0.5%)
Non DFS Used:	0 B
DFS Remaining:	233.14 GB (93%)
Block Pool Used:	1.24 GB (0.5%)
DataNodes usages% (Min/Median/Max/stdDev):	0.25% / 0.57% / 0.66% / 0.18%
Live Nodes	3 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	Tue Sep 28 16:19:45 +0800 2021
Last Checkpoint Time	Wed Sep 29 09:19:55 +0800 2021

For MRS 3.*x* or later: Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS**, click **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

  For versions earlier than MRS 3.*x*: Log in to the MRS console and choose **Components** > **HDFS** > **NameNode (Active)**. Click **DataNodes** and check the number of blocks of all DataNodes that report alarms.



Node	Http Address	Last contact	Capacity	Blocks	Block pool used	Version
node-ana-coreqIIX.34bf0d99-9d47-4e53-a911-d730af226491.com:9866 .39:9866)		0s	83.56 GB	229	568.74 MB (0.66%)	2.8.3-mrs-1.9.0
node-ana-corewAIG.34bf0d99-9d47-4e53-a911-d730af226491.com:9866 (.64:9866)		0s	83.56 GB	216	217.68 MB (0.25%)	2.8.3-mrs-1.9.0
node-ana-corewAlm.34bf0d99-9d47-4e53-a911-d730af226491.com:9866 (.28:9866)		1s	83.56 GB	219	488.4 MB (0.57%)	2.8.3-mrs-1.9.0

Showing 1 to 3 of 3 entries

Previous 1 Next

  For MRS 3.*x* or later: Log in to FusionInsight Manager, choose **Cluster** > *Name of the desired 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.

9.15.4 Changing the DataNode Storage Directory

Scenario

NOTE

This section applies to MRS 3.*x* or later.

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** has been created.
- 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

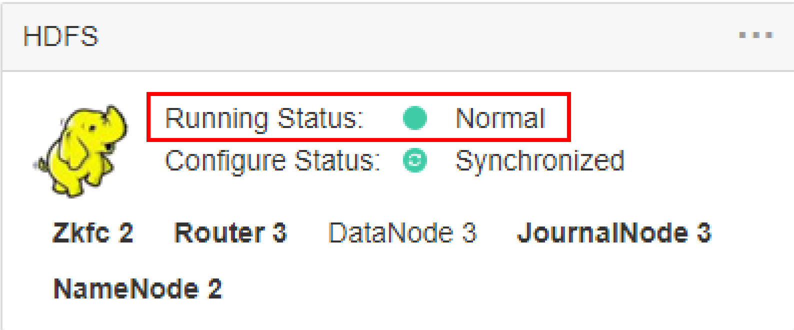
Step 3 Run the following command on the HDFS client to check whether all directories and files in the HDFS root directory are normal:

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** > **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** > **Services** > **HDFS** and click **Stop Service** to stop the HDFS service.

Step 8 Log in as user **root** to each node on which the HDFS service is installed, and perform the following operations:

1. Create a target directory (**data1** and **data2** are original directories in the cluster).

   For example, if the target directory is **${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, if the new directory is **${BIGDATA_DATA_HOME}/hadoop/data3/dn**, run the following command:

chmod 700 ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R

chown omm:wheel ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R

4. Copy the data to the target directory.

   For example, if the original 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** > **Services** > **HDFS** and click **Configurations** then **All Configurations** to access 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**. On the **Cluster** > **Services** page, start each stopped service in the cluster.

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.

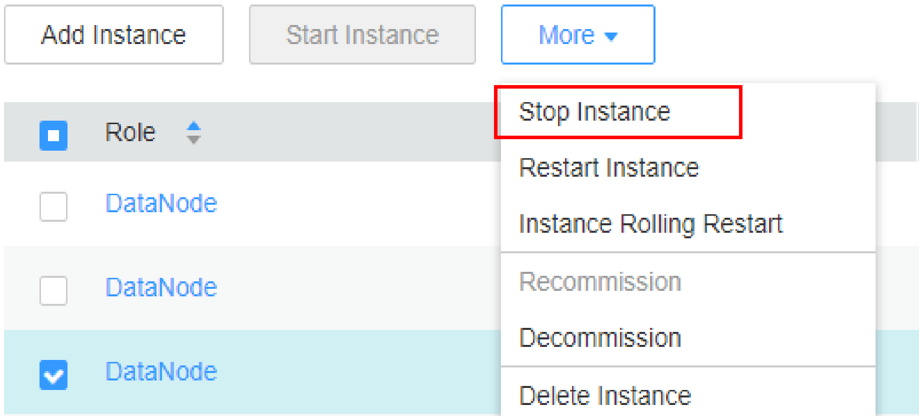
  ```
  The filesystem under path '/' is HEALTHY
  ```

- If other information is displayed, some files are lost or damaged. Check whether **8.4** is correct and run the following command:

  hdfs fsck *Name of the damaged file* **-delete**

Changing the storage directory of a single DataNode instance

Step 12 Choose **Cluster** > **Services** > **HDFS** and click **Instances**. Select the DataNode whose storage directory needs to be modified, click **More**, and select **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, if the target directory is **${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, if the new directory is **${BIGDATA_DATA_HOME}/hadoop/data3/dn**, run the following command:

   chmod 700 ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R

   chown omm:wheel ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R

4. Copy the data to the target directory.

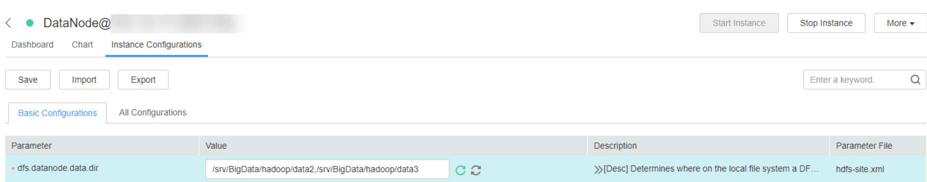
   For example, if the original directory is **${BIGDATA_DATA_HOME}/hadoop/data1/dn** and the target directory is **${BIGDATA_DATA_HOME}/hadoop/data3/dn**, run the following command:

   cp -af ${BIGDATA_DATA_HOME}/hadoop/data1/dn/* ${BIGDATA_DATA_HOME}/hadoop/data3/dn

Step 14 On FusionInsight Manager, choose **Cluster** > **Services** > **HDFS** and click **Instance**. Click the specified DataNode instance and go to the **Configurations** tab 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

9.15.5 Adjusting Parameters Related to Damaged DataNode Disk Volumes

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 9-38 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.	Versions earlier than MRS 3.x: 0 MRS 3.*x* or later: –1

9.15.6 Configuring the Maximum Lifetime of an HDFS 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 9-39 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

9.15.7 Using DistCp to Copy HDFS Data Across Clusters

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)
- If data needs to be copied between clusters, the data copy function must be enabled for both the 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. Usage of the update and overwrite options is as follows:
   - **-update** copies files from source that do not exist at the target or differ from the target version.
   - **-overwrite** overwrites target files that exist at the target.

   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 9-40 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

Option	Description
-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 accelerates DistCp execution.
-overwrite	File at the target location that is to be overwritten
-update	A file at the target location will be updated if it has different size and checksum from 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 will not be copied.
-async	Run **distcp** asynchronously.
-atomic {-tmp <tmp_dir>}	Perform an atomic copy. You can add a temporary directory during copying.
-bandwidth	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 that redundant files at the target location are deleted.
-diff <oldSnapshot> <newSnapshot>	The differences between the old and new versions are copied to 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}	Copy policy. The default policy is **uniformsize**, that is, each task copies the same number of bytes.

FAQs of DistCp

- To avoid timeouts when copying large files with DistCp, adjust the MapReduce timeout period for 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.*
```

- 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**

- 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
```

9.15.8 Configuring the NFS Server to Store NameNode Metadata

Scenario

NOTE

This section applies to MRS 3.*x* or later.

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 node 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** > **Services** > **HDFS**. Click **Configurations** then **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

9.16 Common Issues About HDFS

9.16.1 What Should I Do If an Error Is Reported When I Run DistCp Commands?

Question

DistCp commands fail to run in a security cluster and exceptions are thrown.

The following client exception is reported:

```
Invalid arguments:Unexpected end of file from server
```

The following server exception is reported:

```
javax.net.ssl.SSLException:Unrecognized SSL message, plaintext connection?
```

Answer

When a user uses **webhdfs://** in a DistCp command, the preceding exceptions are thrown because the cluster uses HTTPS, that is, the **dfs.http.policy** value in the **hdfs-site.xml** file configured in *Client installation directory***/HDFS/hadoop/etc/hadoop** is **HTTPS_ONLY**. To avoid this exception, replace **webhdfs://** with **swebhdfs://**.

For example:

./hadoop distcpswebhdfs://IP:PORT/testfile hdfs://IP:PORT/testfile1

9.16.2 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.

9.16.3 "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.

9.16.4 What Should I Do If the HDFS Web UI Cannot Update the Information About the Damaged Data?

Question

1. When an error occurs in the directory specified by **dfs.datanode.data.dir** of DataNode due to permission issue or disk damage, the HDFS web UI does not display information about the damaged data.
2. After the fault is rectified, the HDFS web UI does not remove the information about the damaged data in a timely manner.

Answer

1. When an error occurs during file operations, the DataNode checks the disk status. If data is damaged, the DataNode notifies the NameNode, which then displays the issue on the HDFS web UI.
2. After the fault is rectified, restart the DataNode. When the DataNode is restarted, the system checks the status of all data and uploads the damaged data to the NameNode. The damaged data information will not be displayed only after the DataNode is restarted.

9.16.5 What Should I Do If the HDFS Client Is Irresponsive 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 irresponsive. The HDFS clients that are connected to the overloaded NameNode fail to respond and further operations cannot be performed. However, the HDFS clients that are newly connected to the NameNode will be switched to a backup NameNode and run properly.

Answer

When the preceding error occurs, the default configuration was used (as described in **Table 9-41**): the keep alive mechanism is enabled for the RPC connection between the HDFS client and the NameNode. The keep alive mechanism keeps the HDFS client waiting for server's responses and prevents the connection from being out timed, causing the irresponsive HDFS client.

In this case, you can:

- 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.

Solution:

To avoid this problem, add the following configurations to *Client installation path*/**HDFS/hadoop/etc/hadoop/core-site.xml**.

Table 9-41 Parameters

Parameter	Description	Default Value
ipc.client.ping	If this parameter is **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 this parameter is **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 irresponsiveness of HDFS when the NameNode is overloaded for a long time, set this parameter to **false**.	true
ipc.ping.interval	If **ipc.client.ping** is **true**, this parameter indicates the interval between sending the ping messages. If **ipc.client.ping** is **false**, this parameter indicates the connection timeout interval. To avoid the irresponsiveness of HDFS when the NameNode is overloaded for a long time, you are advised to set this parameter to a large value, for example 900000 (ms) to avoid timeout when the server is busy.	60000

9.16.6 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.

9.16.7 Why Does DataNode Fail to 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.

9.16.8 Can I Modify the DataNode Data Storage Directory?

Question

- In DataNode, the storage directory of data blocks is specified by **dfs.datanode.data.dir**. Can I modify the parameter to change the data storage directory?
- Can I copy files from the data storage directory?

Answer

During the system installation, you need to configure the **dfs.datanode.data.dir** parameter to specify one or multiple root directories.

- Exercise caution when modifying **dfs.datanode.data.dir** since you may add a new root directory.
- Do not delete the original storage directory. Otherwise, data blocks will lose and files cannot be read or write.
- Do not delete or modify data blocks in the storage directory. Otherwise, data blocks may be lost.

NOTE

NameNode and JournalNode have similar configuration items. Do not delete the original storage directory or delete or modify data blocks in the storage directory.

- dfs.namenode.edits.dir
- dfs.namenode.name.dir
- dfs.journalnode.edits.dir

9.16.9 What Can I Do If the DataNode Capacity Is Incorrectly Calculated?

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.

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>
```

9.16.10 Why Is Data in the Cache Lost When Small Files Are Stored?

Question

The system is powered off when it is saving small files. As a result, the data in the cache is lost.

Answer

Blocks in the cache were not written to the disk immediately due to the power failure. To synchronously write the cached blocks to the disk, set **dfs.datanode.synconclose** to **true** in *Client installation path***/HDFS/hadoop/etc/hadoop/hdfs-site.xml**.

By default, **dfs.datanode.synconclose** is set to **false**. Although the performance is high, data stored in the cache will be lost after a power failure. You can set **dfs.datanode.synconclose** to **true** to solve this problem. However, the performance will be greatly affected. Set this parameter based on the application scenario.

9.16.11 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 deployed has the RAM disk, but the value of **dfs.datanode.max.locked.memory** is not set or is set to a value less than **dfs.blocksize**, the first replica is written into the disk of the DataNode where the client is deployed, and other replicas are written into the disks of other nodes. (To check the parameter value, log in to FusionInsight Manager and choose **Cluster** > **Services** > **HDFS**. On the displayed page, click **Configurations** > **All Configurations**, and search for the parameter.)

9.16.12 Why Some Blocks Are Missing on the NameNode UI?

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 upgrade:

   Client A writes some data to file X. (Assume A bytes are written.)

2. During upgrade:

   Client A still writes data to file X. (The data in the file is A + B bytes.)

3. After upgrade:

   Client A completes the file writing. The final data is A + B bytes.

4. Rollback:

   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.

9.17 HDFS Troubleshooting

9.17.1 Why Is "java.net.SocketException" 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 9-14 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
```

9.17.2 It Takes a Long Time to Restart NameNode After a Large Number of Files Are Deleted

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.

9.17.3 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_0000000000013259401-00000000000
-rw-------. 1 omm wheel     575 Oct  2 01:13 edits_0000000000013259409-00000000000
-rw-------. 1 omm wheel      42 Oct  2 01:13 edits_0000000000013259417-00000000000
-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  576 Oct  2 00:41 edits_0000000000013259151-0000000000013259158
-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
-rw-------. 1 omm wheel  983 Oct  2 01:06 edits_0000000000013259345-0000000000013259356
```

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.

9.17.4 The standby NameNode Fails to Be Started After It Is Powered Off During Metadata 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_000000000000096
080,
cpktTxId=0000000000000096080) | FSImage.java:685
java.io.IOException: Invalid MD5 file /srv/BigData/namenode/current/f
simage_0000000000000096080.md5:
the content "[illegible] 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.

9.17.5 DataNode Fails to Be Started When the Number of Disks Defined in dfs.datanode.data.dir Equals the Value of dfs.datanode.failed.volumes.tolerated

Question

When the number of disks defined in **dfs.datanode.data.dir** equals the value of **dfs.datanode.failed.volumes.tolerated**, DataNode fails to be started.

Answer

By default, if a single disk is faulty, the HDFS DataNode process is stopped. As a result, NameNode schedules extra copies for each block stored in DataNode, causing block replication on normal disks.

To prevent this problem, you can configure a DataNodes tolerance value for the **dfs.data.dir** fault. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **HDFS**. On the displayed page, click **Configurations** > **All Configurations**, and search for **dfs.datanode.failed.volumes.tolerated**. For example, if this parameter is set to **3**, DataNode startup fails only when four or more directories are faulty.

To prevent DataNode faults, the value of **dfs.datanode.failed.volumes.tolerated** must be less than the number of configured volumes. You can also set **dfs.datanode.failed.volumes.tolerated** to **-1**, which is equivalent to **n-1** (**n** indicates the number of volumes). This way, DataNode will be started normally.

9.17.6 "ArrayIndexOutOfBoundsException: 0" Occurs When HDFS Invokes getsplit of FileInputFormat

Question

When HDFS invokes the **getSplit** method of **FileInputFormat**, "ArrayIndexOutOfBoundsException: 0" is displayed. The log is as follows:

```
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 rack information of each block is in the format of /default/rack0/:,/default/rack0/datanodeip:port.

Blocks are damaged or lost. As a result, the IP address and port number of the host corresponding to the blocks are empty. To handle this problem, use **hdfs fsck** to check the health status of the file blocks, delete the damaged or lost blocks, and run task again.

10 Using Hive

10.1 Hive User Permission Management

10.1.1 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.
- MRS 3.*x* or later supports Ranger. 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.

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 10-1 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 10-2 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 if Kerberos authentication is disabled for the cluster (normal mode).	If Kerberos authentication is disabled, Ranger authentication is disabled by default. To enable Ranger authentication, add the following restrictions: ● Whitelist: Only whitelisted parameters can be configured 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 operation is not recommended cause it poses security risks. ● Forbid the execution of reflect, reflect2, java_method, and in_file functions. If the functions are required by services, add **hive.server2.builtin.udf.blacklist** to the HiveServer custom parameters, and set it to **mpty_blacklist** to allow running these functions on Hive. This operation is not recommended cause it poses security risks.

10.1.2 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** for MRS 3.*x* or later that supports Ranger.

Prerequisites

- The MRS cluster administrator has understood service requirements.
- Log in to FusionInsight Manager.
- The Hive client has been installed.

Procedure

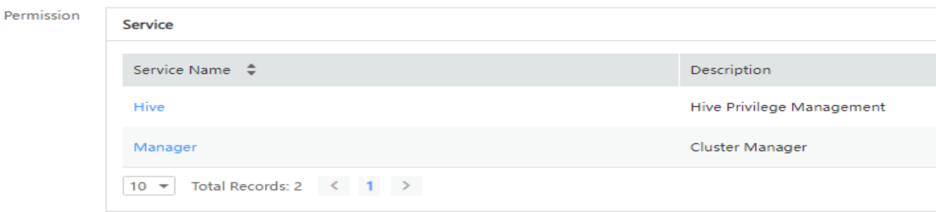
For versions earlier than MRS 3.x, perform the following operations to create a Hive role:

Step 1 Log in to MRS Manager.

Step 2 Choose **System** > **Permission** > **Manage Role**.

Step 3 Click **Create Role**, and set **Role Name** and **Description**.

Step 4 Set permissions. For details, see **Table 10-3**.



- **Hive Admin Privilege**: Hive administrator permissions. If you want to use this permission, run the **set role admin** command to set the permission before running SQL statements.
- **Hive Read Write Privileges**: Hive data table management permission, which is the operation permission to set and manage the data of created tables. Select the permissions of a database as required. To specify permissions on tables, click the database name and select the permissions of the 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 10-3 Setting a role

<table>
<tr><th>Scenario</th><th>Role Authorization</th></tr>
<tr><td>Setting the Hive administrator permission</td><td>In the Permission table, click 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 client is installed.
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;</td></tr>
<tr><td>Setting the permission to query a table of another user in the default database</td><td>1. In the Permission table, choose Hive > Hive Read Write Privileges.
2. In the Permission column of the specified table, select SELECT.</td></tr>
<tr><td>Setting the permission to query a table of another user in the default database</td><td>1. In the Permission table, choose Hive > Hive Read Write Privileges.
2. In the Permission column of the specified table, select Insert.</td></tr>
<tr><td>Setting the permission to import data to a table of another user in the default database</td><td>1. In the Permission table, choose Hive > Hive Read Write Privileges.
2. In the Permission column of the specified table, select Delete and Insert.</td></tr>
</table>

Scenario	Role Authorization
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 **Yarn** > **Scheduler Queue** > **root**. 2. In the **Permission** column of the default queue, select **Submit**.

Step 5 Click **OK**, and return to the **Role** page.

Step 6 Choose **System** > **Manage User** > **Create User**.

Step 7 Enter the username, set **User Type** to **Human-machine**, set the user password, add a user group bound with the Hive administrator role, bind the new Hive role to the user group, and click **OK**.

Step 8 After the user is created, you can run the SQL statement using the user.

----End

For MRS 3.*x* or later, perform the following operations to create a Hive role:

Step 1 Log in to FusionInsight Manager. For details, see**Accessing FusionInsight Manager (MRS 3.x or Later)**

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 10-4**.

- 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

- In MRS 3.1.0, Hive role management supports the Hive administrator permission, and the permissions of accessing tables and views, without granting the database permission.
- For MRS 3.1.2 or later, 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 10-4 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;**

Task	Role Authorization
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**.
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.

Step 6 Choose **Permission** > **User**. On the displayed page, click **Create User**.

Step 7 Enter the username, set **User Type** to **Human-machine**, set the user password, add a user group with Hive permissions, select the primary group, bind the new role to the user group, and click **OK**.

Step 8 Run SQL statements as the created user.

----End

10.1.3 Granting Hive User 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.
- MRS 3.x or later supports Ranger. 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

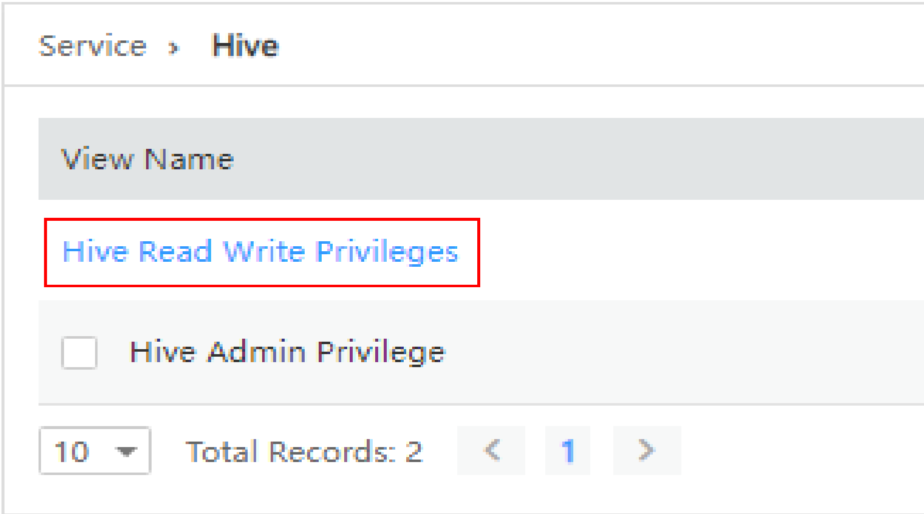
  Users have complete permission on the tables created by themselves in Hive and the HDFS. To access the tables created by others, they need to be granted the permission. After the Hive metadata permission is granted, the HDFS permission is automatically granted. The procedure for granting a role the permission of querying, inserting, and deleting **htable** data is as follows:

For versions earlier than MRS 3.x, perform the following operations to grant table permissions:

a. On MRS Manager, choose **System** > **Permission** > **Manage Role**.

b. Locate the row that contains **hrole**, and click **Modify**.

c. Choose **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.

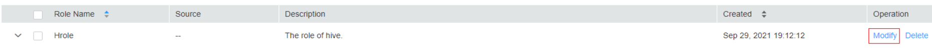
e. In the **Permission** column of the **htable** table, select **Select**, **Insert**, and **Delete**.

f. Click **OK**.

For MRS 3.*x* and later, perform the following operations to grant table permissions:

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:

  For versions earlier than MRS 3.x, perform the following operations to grant column permissions:

  a. On MRS Manager, choose **System** > **Permission** > **Manage Role**.
  b. Locate the row that contains **hrole**, and click **Modify**.
  c. Choose **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**.

  For MRS 3.*x* and later, perform the following operations to grant column permissions:

  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.

For versions earlier than MRS 3.x, perform the following database authorization operations:

a. On MRS Manager, choose **System** > **Permission** > **Manage Role**.
b. Locate the row that contains **hrole**, and click **Modify**.
c. Choose **Hive** > **Hive Read Write Privileges**.
d. In the **Permission** column of the **hdb** database, select **Select** and **Create**.
e. Click **OK**.

For MRS 3.*x* and later, perform the following operations to grant database permissions:

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.

Concepts

Table 10-5 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

Scenario	Required Permission
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

10.1.4 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.
- MRS 3.*x* or later supports Ranger. 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 in MRS Earlier than 3.x

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 MRS Manager.

Step 2 In the **Permission** table, choose **Yarn** > **Scheduler Queue** > **root**.

Step 3 In the **Permission** column of the default queue, select **Submit** and click **OK**.

Step 4 In the **Permission** table, choose **Hive** > **Hive Read Write Privileges** > **default**, select **Select** for **thc**, and click **OK**.

----End

Association with Yarn in MRS 3.*x* or Later

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 in MRS Earlier than 3.x

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 MRS Manager, create an HBase role, for example, **hive_hbase_create**, and grant the permission to create HBase tables.

In the **Permission** table, choose **HBase** > **HBase Scope** > **global**, select **create** of the namespace **default**, and click **OK**.

Step 2 On MRS 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 Log in to the node where the client is installed.

Step 4 Run the following command to configure environment variables:

source /opt/client/bigdata_env

Step 5 Run the following command to authenticate the user:

kinit hbase_creates_user

Step 6 Run the following command to go to the shell environment of the Hive client:

beeline

Step 7 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 8 On the role management page of MRS 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 **Permission** table, choose **HBase** > **HBase Scope** > **global** > **default**, select **Read** for the **thh** table, and click **OK** to grant the HBase role the permission to query the table.
2. Edit a role. In the **Permission** table, choose **HBase** > **HBase Scope** > **global** > **hbase**. Select **Execute** for **hbase:meta**, and click **OK**.
3. Edit a role. In the **Permission** table, choose **Hive** > **Hive Read Write Privileges** > **default**, select **Select** for **thh**, and click **OK**.

Step 9 On MRS 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 10 Run the following command to configure environment variables:

source /opt/client/bigdata_env

Step 11 Run the following command to authenticate users:

kinit hbase_select_user

Step 12 Run the following command to go to the shell environment of the Hive client:

beeline

Step 13 Run the following command to use an HQL statement to query HBase table data:

select * from thh;

----End

Hive over HBase Authorization in MRS 3.x and Later

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

10.2 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 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 (Versions Earlier Than MRS 3.x)

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*

NOTE

After a beeline connection is established, you can compile and submit HQL statements to execute related tasks. To run the Catalog client command, you need to run the **!q** command first to exit the beeline environment.

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

Using the Hive Client (MRS 3.*x* or Later)

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 10-6 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.

Command	Description
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.
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 Ivy 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.

10.3 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 10-7 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 Download the client configuration file.

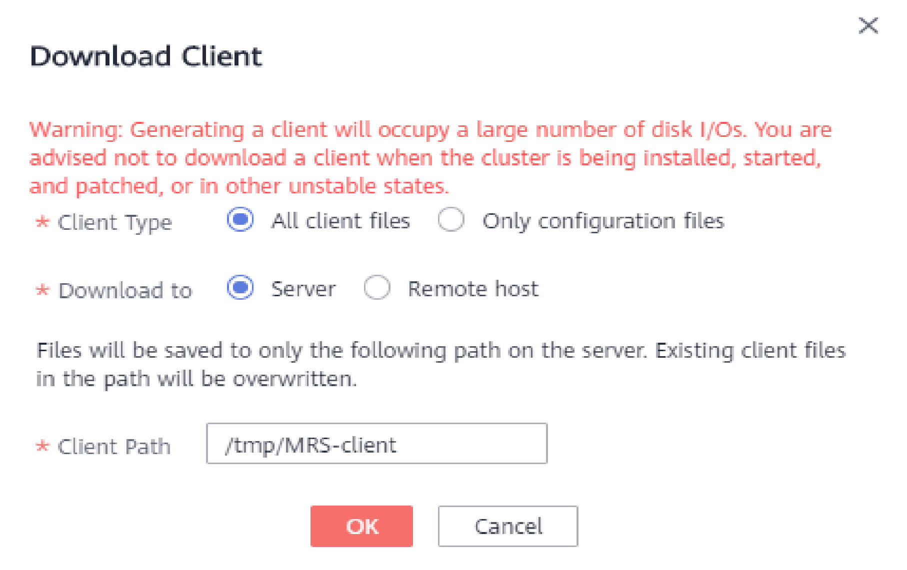
For versions earlier than MRS 3.x, perform the following operations:

1. Log in to MRS Manager. For details, see **Accessing Manager**. Then, choose **Services**.

2. Click **Download Client**.

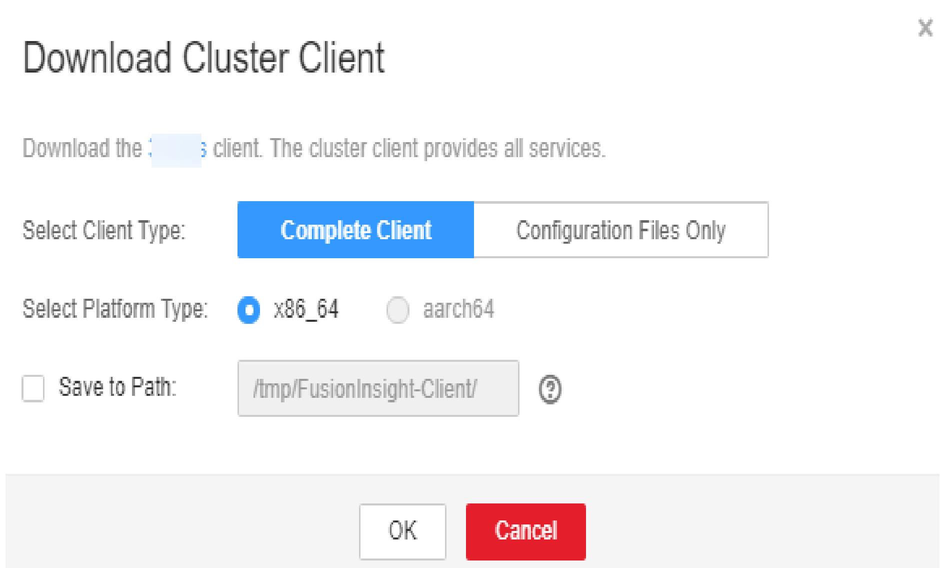
   Set **Client Type** to **Only configuration files**, **Download to** to **Server**, and click **OK** to generate the client configuration file. The generated file is saved in the **/tmp/MRS-client** directory on the active management node by default.

   Figure 10-1 Downloading only the client configuration files



For MRS 3.x and later, perform the following operations:

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Dashboard** > **More** > **Download Client**.
3. 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.

For versions earlier than MRS 3.x, perform the following operations:

1. On the MRS console, choose **Active Clusters** and click a cluster name. In the **Nodes** tab, view the node names. The node whose name contains **master1** is the Master1 node, and the node whose name contains **master2** is the Master2 node.

   The active and standby management nodes of MRS Manager are installed on Master nodes by default. Because Master1 and Master2 are switched over in active and standby mode, Master1 is not always the active management node of MRS Manager. Run a command in Master1 to check whether Master1 is active management node of MRS Manager. For details about the command, see **Step 2.4**.
2. Log in to the Master1 node using the password as user **root**.
3. Run the following commands to switch to user **omm**:

   sudo su - root

   su - omm
4. Run the following command to check the active management node of MRS Manager:

   sh ${BIGDATA_HOME}/om-0.0.1/sbin/status-oms.sh

   In the command output, the node whose **HAActive** is **active** is the active management node, and the node whose **HAActive** is **standby** is the standby management node. In the following example, **mgtomsdat-sh-3-01-1** is the active management node, and **mgtomsdat-sh-3-01-2** is the standby management node.

```
Ha mode
double
NodeName              HostName                      HAVersion          StartTime                HAActive
HAAllResOK            HARunPhase
192-168-0-30          mgtomsdat-sh-3-01-1           V100R001C01        2014-11-18 23:43:02
active                normal               Actived
192-168-0-24          mgtomsdat-sh-3-01-2           V100R001C01        2014-11-21 07:14:02
standby               normal               Deactived
```

5. Log in to the active management node as user **root**, for example, node **192-168-0-30**.

For MRS 3.x and later, perform the following operations:

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 go to the client installation directory:

cd /opt/client

The cluster client has been installed in advance. The following client installation directory is used as an example. Change it based on the site requirements.

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, for example, **192-168-0-30**, run the following command to switch to the client directory, for example, **/opt/client**.

   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:

   kinit *MRS cluster user*

   Example: user **kinit hiveuser**

   The current user must have the permission to create Hive tables. If Kerberos authentication is disabled, skip this step.

4. Run the client command of the Hive component directly.

   beeline

Step 6 Run the Hive client command to implement service A.

Operations on internal tables:

1. Create the **user_info** user information table according to **Table 10-7** and add data to it.

   create table user_info(id string,name string,gender string,age int,addr string);

   For MRS 1.x or MRS 3.x and later, perform the following operations:

insert into table user_info(id,name,gender,age,addr) values("12005000201","A","Male",19,"City A");

2. 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);

3. 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';

4. Delete the user information table.

   drop table user_info;

Operations on external partition tables:

Create an external partition table and import data.

1. Create a path for storing external table data.

   hdfs dfs -mkdir /hive/

   hdfs dfs -mkdir /hive/user_info

2. 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.

3. Import data.

   a. Execute the insert statement to insert data.

      insert into user_info partition(year="2018") values ("12005000201","A","Male",19,"City A");

   b. Run the **load data** command to import file data.

      i. Create a file based on the data in **Table 10-7**. 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. Upload the file to HDFS.

         hdfs dfs -put txt.log /tmp

      iii. Load data to the table.

         load data inpath '/tmp/txt.log' into table user_info partition (year='2011');

4. Query the imported data.

   select * from user_info;

5. Delete the user information table.

   drop table user_info;

6. Run the following command to exit:

   !q

----End

10.4 Configuring Hive Data Storage and Encryption

10.4.1 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.

NOTE

This section applies to MRS 3.*x* or later.

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

10.4.2 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.

NOTE

This feature applies only to MRS 3.1.2 or later.

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 a specified database

  show frozen tables in *Database name*;

Querying Frozen Partitions of a Frozen Table

Querying frozen partitions

show frozen partitions *Table name*;

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.

10.4.3 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.

NOTE

This feature applies only to MRS 3.1.2 or later.

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.

10.4.4 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

After importing data from a common Hive table to a Hive column encryption table, 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

10.5 Hive on HBase

10.5.1 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

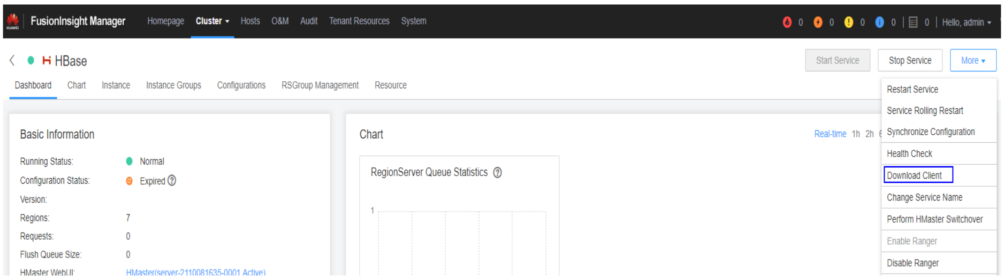
The mutual trust relationship has been configured between the two security clusters with Kerberos authentication enabled.

Procedure for Configuring Hive on HBase Across Clusters

Step 1 Download the HBase configuration file and decompress it.

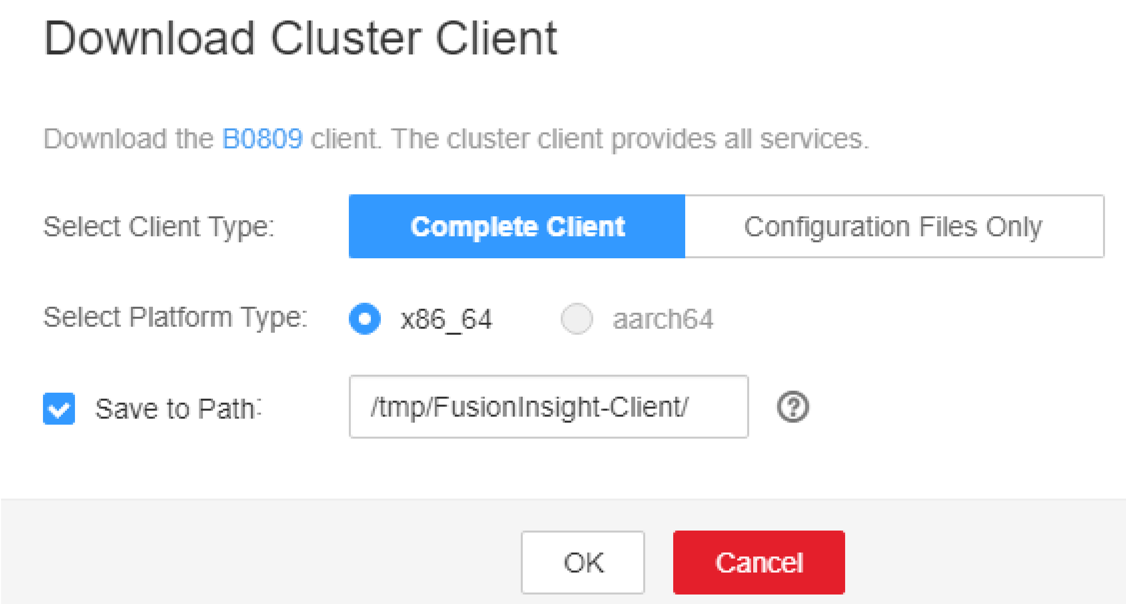
1. Log in to FusionInsight Manager of the target HBase cluster, click **Cluster** and choose **Services** > **HBase**.
2. Choose **More** > **Download Client**.

Figure 10-2 Downloading the HBase client



3. Download the HBase configuration file and choose **Configuration Files only** for **Select Client Type**.

Figure 10-3 Downloading the HBase configuration file



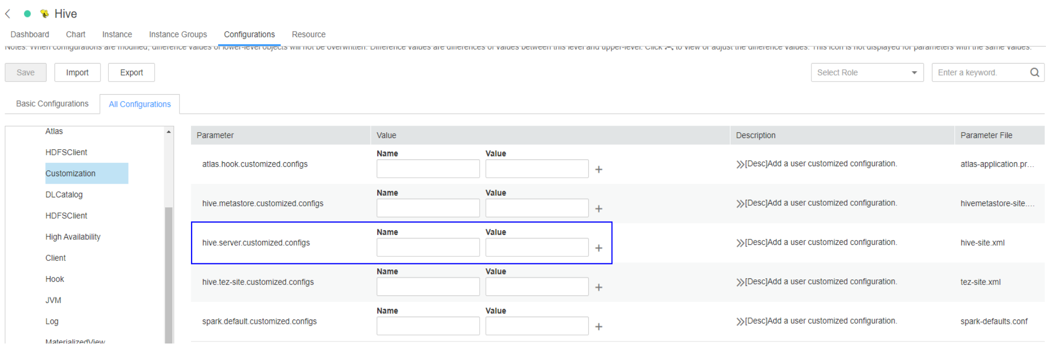
Step 2 Log in to FusionInsight Manager of the source Hive cluster.

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 10-4 Custom configurations of the HiveServer role



Step 4 Save the configurations and restart Hive.

----End

10.5.2 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 10-8 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

10.6 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, the following relational databases can use Hive to read data:

- DB2
- Oracle

NOTE

This section applies to MRS 3.*x* or later.

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.

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.

```
-- 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. 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.
"qubit.sql.dbcp.password" = "xxx");
```

----End

10.7 Enterprise-Class Enhancement of Hive

10.7.1 Configuring Automatic Removal of Old Data in the Hive Directory to the Recycle Bin

Scenario

This function applies to Hive.

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.

Step 1 The Hive service configuration page is displayed.

- For versions earlier than MRS 3.x, click the cluster name. On the cluster details page that is displayed, choose **Components** > **Hive** > **Service Configuration**, and select **All** from the **Basic** drop-down list.

  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.)

- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. And choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **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**. Restart all Hive instances after the modification.

----End

10.7.2 Configuring Hive to Insert Data to a Directory That Does Not Exist

Scenario

This function applies to Hive.

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.

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**.

NOTE

The parameter adjustment of this function is the same as that of the custom parameters added in **Configuring Automatic Removal of Old Data in the Hive Directory to the Recycle Bin**.

Procedure

Step 1 The Hive service configuration page is displayed.

- For versions earlier than MRS 3.x, click the cluster name. On the cluster details page that is displayed, choose **Components** > **Hive** > **Service Configuration**, and select **All** from the **Basic** drop-down list.

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.)

- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. And choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **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**. Restart all Hive instances after the modification.

----End

10.7.3 Forbidding Location Specification When Hive Internal Tables Are Created

Scenario

This function is available in Hive and Spark2x for MRS 3.x or later, or Hive and Spark for versions earlier than MRS 3.x.

After this function is enabled, the **location** keyword cannot be specified when a Hive internal table is created. Specifically, after a table is created, the table path following the location keyword is created in the default **\warehouse** directory and cannot be specified to another directory. If the location is specified when the internal table is created, 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 The Hive service configuration page is displayed.

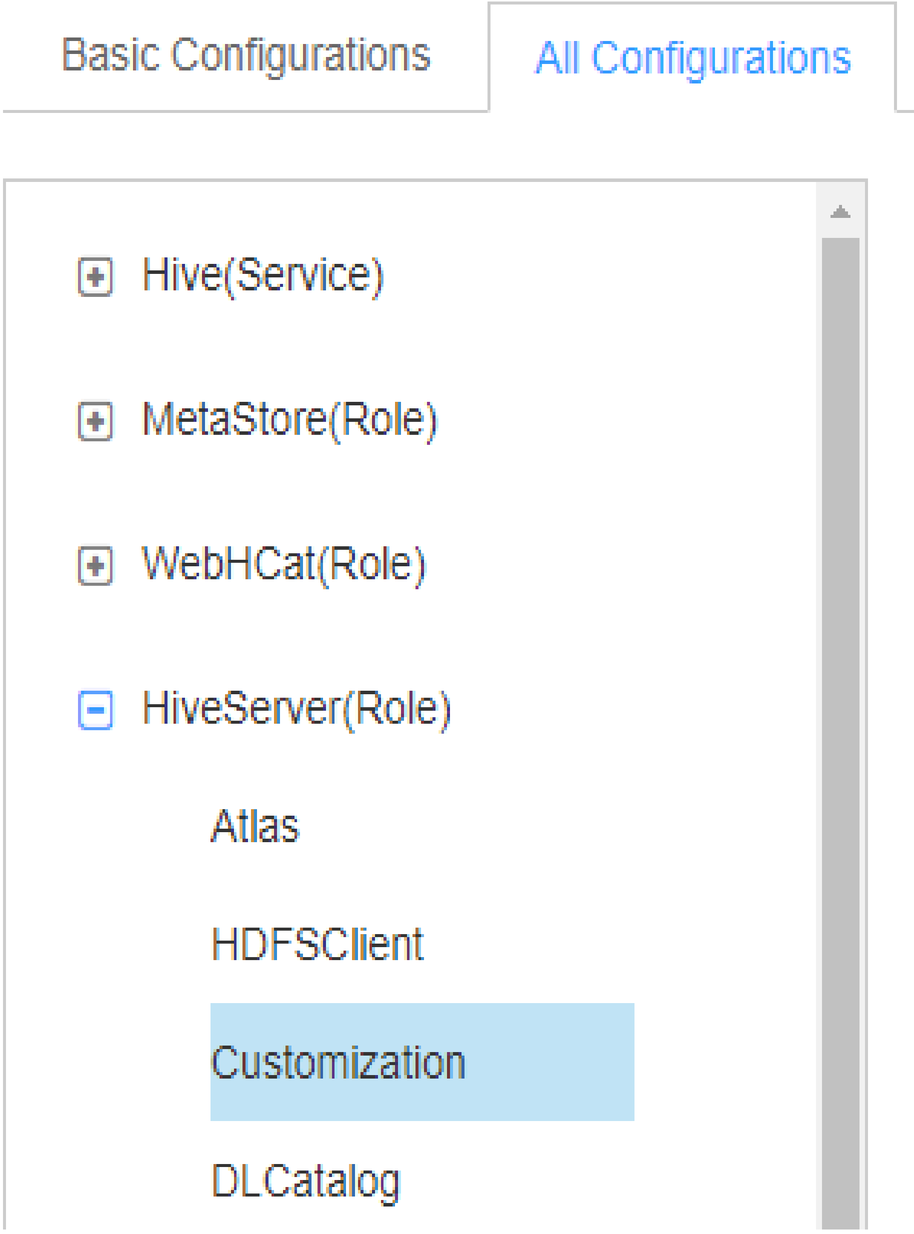
- For versions earlier than MRS 3.x, click the cluster name. On the cluster details page that is displayed, choose **Components** > **Hive** > **Service Configuration**, and select **All** from the **Basic** drop-down list.

  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.)

- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. And choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **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**. Restart all Hive instances after the modification.



Step 3 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

10.7.4 Creating a Foreign Table in a Directory (Read and Execute Permission Granted)

Scenario

This function is available in Hive and Spark2x for MRS 3.x or later, or Hive and Spark for versions earlier than MRS 3.x.

After this function is enabled, the user or user group that has the read and execute permissions on a directory can create foreign tables in the directory without checking whether the current user is the owner of the directory. In addition, the directory of a foreign table cannot be stored in the default directory **\warehouse**. In addition, do not change the permission of the directory during foreign table authorization.

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 The Hive service configuration page is displayed.

- For versions earlier than MRS 3.x, click the cluster name. On the cluster details page that is displayed, choose **Components** > **Hive** > **Service Configuration**, and select **All** from the **Basic** drop-down list.

  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.)

- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. And choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **Configurations** > **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**. Restart all Hive instances after the modification.

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

10.7.5 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 The Hive service configuration page is displayed.

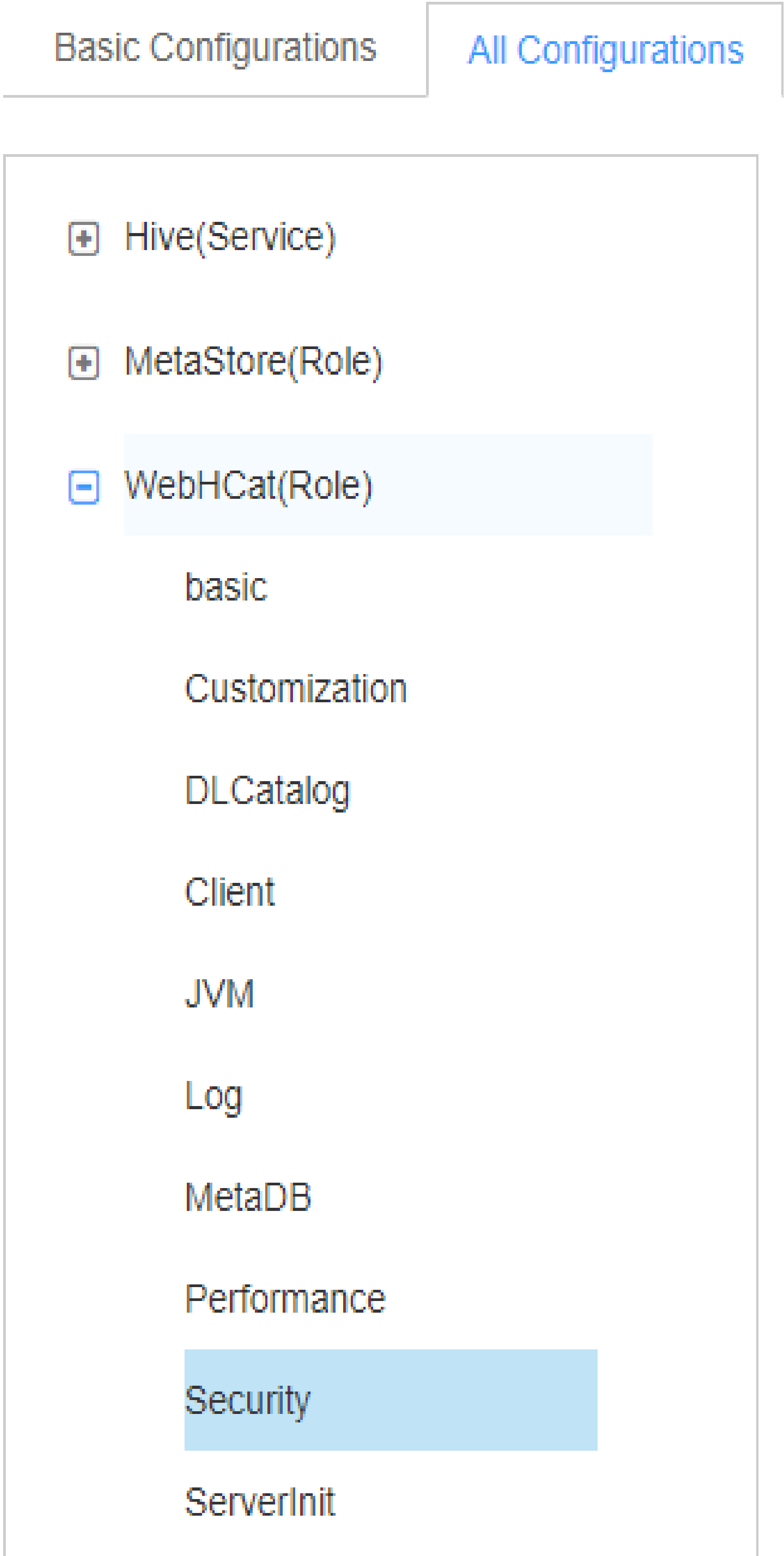
- For versions earlier than MRS 3.x, click the cluster name. On the cluster details page that is displayed, choose **Components** > **Hive** > **Service Configuration**, and select **All** from the **Basic** drop-down list.

  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.)
- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. And choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **Configurations** > **All Configurations**.

Step 2 Modify the Hive configuration.

- For versions earlier than MRS 3.x: Enter the parameter name in the search box, search for **templeton.protocol.type**, change the parameter value to **HTTPS** or **HTTP**, and restart the Hive service to use the corresponding protocol.
- For MRS 3.*x* or earlier: 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.


Basic Configurations
All Configurations
Hive(Service)
MetaStore(Role)
WebHCat(Role)
basic
Customization
DLCatalog
Client
JVM
Log
MetaDB
Performance
Security
ServerInit

----End

10.7.6 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

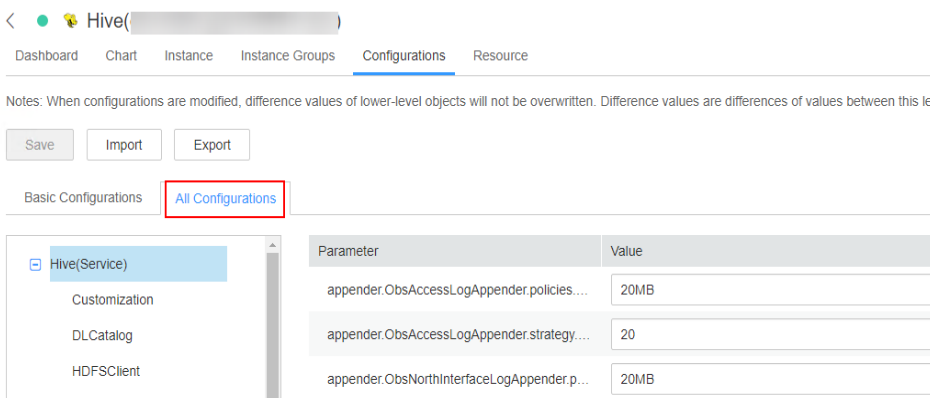
Step 1 The Hive service configuration page is displayed.

- For versions earlier than MRS 3.x, click the cluster name. On the cluster details page that is displayed, choose **Components** > **Hive** > **Service Configuration**, and select **All** from the **Basic** drop-down list.

  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.)

- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. And choose **Cluster** > *Name of the desired 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

10.7.7 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 to switch the engine and enable the **yarn.timeline-service.enabled** parameter:

set hive.execution.engine=tez;

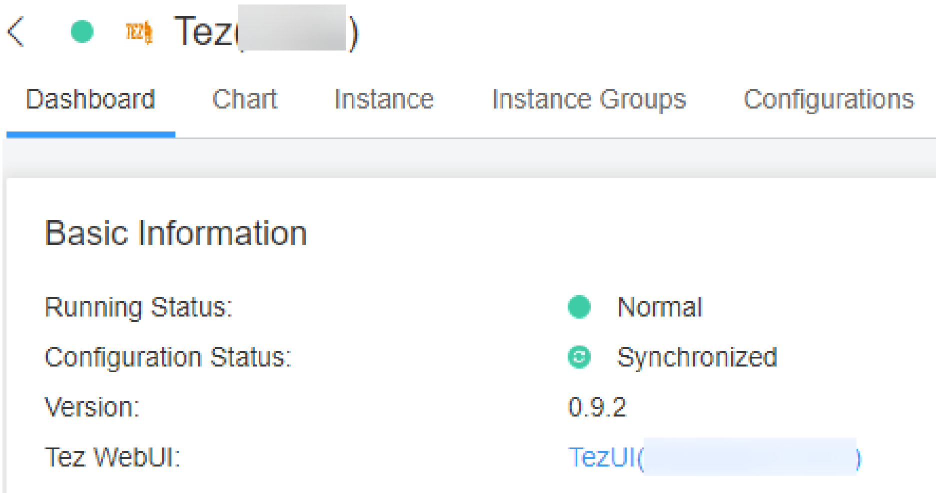
set yarn.timeline-service.enabled=true;

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 Submit and execute the Tez tasks.

Step 4 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Tez** > **TezUI** *(host name)* to view the task execution status on the TezUI page.

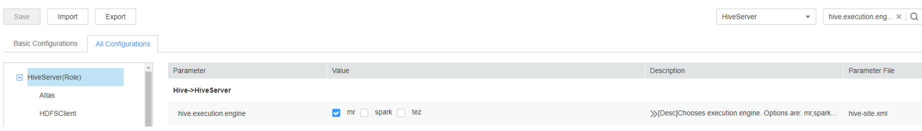


For versions earlier than MRS 3.x, log in to MRS Manager, choose **Services**, and click **Tez**. On the displayed page, click the link next to **Tez WebUI** to view the task execution status on the TezUI page.

----End

Switching the Default Execution Engine of Hive to Tez

Step 1 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **Configurations** > **All Configurations** > **HiveServer(Role)**, and search for **hive.execution.engine**.



For versions earlier than MRS 3.x, log in to MRS Manager, choose **Services**, and click **Hive**. On the displayed page, click the **Service Configuration** tab, select **All** from the **Type** drop-down list. On the navigation pane on the left, choose **HiveServer** 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**.

Step 4 Add custom parameter **yarn.timeline-service.enabled** to **yarn.site.customized.configs** and set it to **true**.

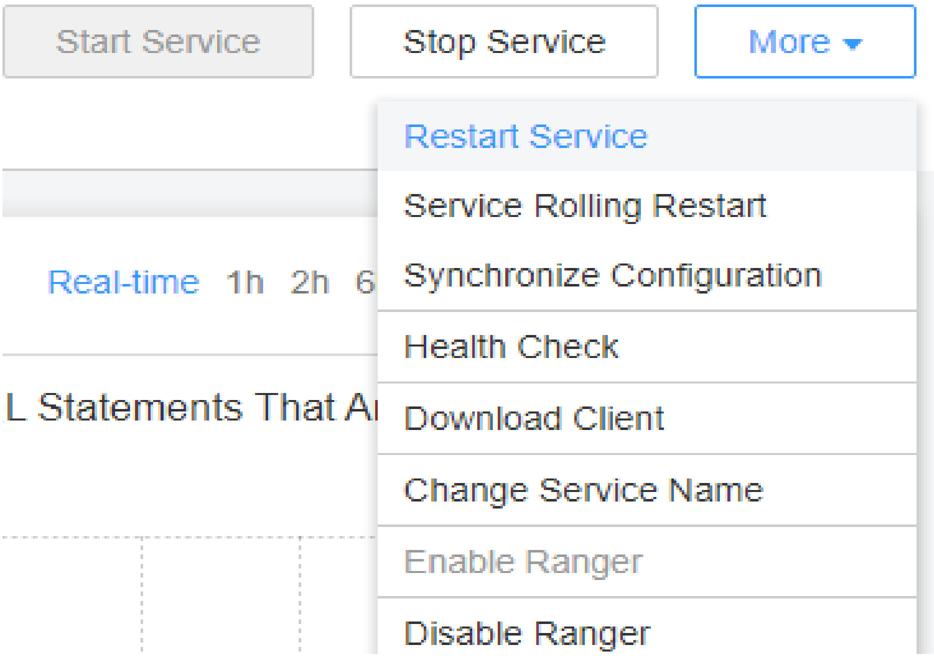
NOTE

- 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 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**.

For versions earlier than MRS 3.x, click **Save Configuration** and click **Yes** in the displayed dialog box.

Step 6 Choose **Dashboard** > **More** > **Restart Service** to restart the Hive service. Enter the password to restart the service.



For versions earlier than MRS 3.x, Click the **Service Status** tab and choose **More** > **Restart Service** 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** > *Name of the desired cluster* > **Services** > **Tez** > **TezUI** *(host name)*. On the displayed TezUI page, view the task execution status.

For versions earlier than MRS 3.x, log in to MRS Manager, choose **Services**, and click **Tez**. On the displayed page, click the link next to **Tez WebUI** to view the task execution status on the TezUI page.

----End

10.7.8 Hive Load Balancing

10.7.8.1 Configuring the Maximum Number of Maps for a Hive Task

Scenario

- This function applies to Hive.
- This function is used to limit the maximum number of maps for Hive tasks on the server to avoid performance deterioration caused by overload of the HiveSever service.

Procedure

Step 1 The Hive service configuration page is displayed.

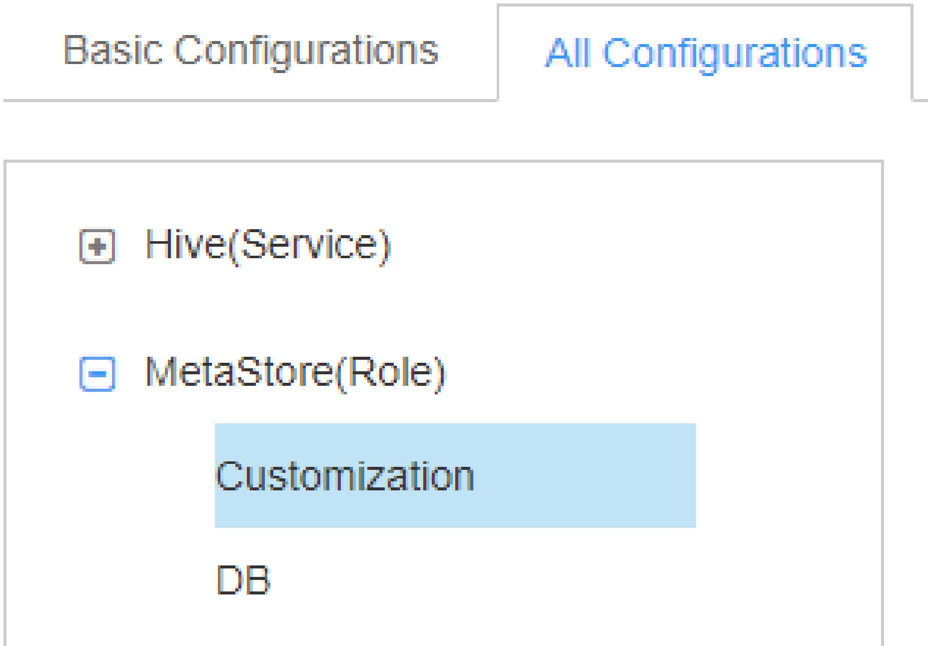
- For versions earlier than MRS 3.x, click the cluster name. On the cluster details page that is displayed, choose **Components** > **Hive** > **Service Configuration**, and select **All** from the **Basic** drop-down list.

  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.)

- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. And choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **Configurations** > **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. Restart all Hive instances after the modification.



----End

10.7.8.2 Configuring User Lease Isolation to Access HiveServer on a Specified Node

Scenario

- This function applies to Hive.
- This function can be enabled to specify specific users to access HiveServer services on specific nodes, achieving HiveServer resource isolation.

NOTE

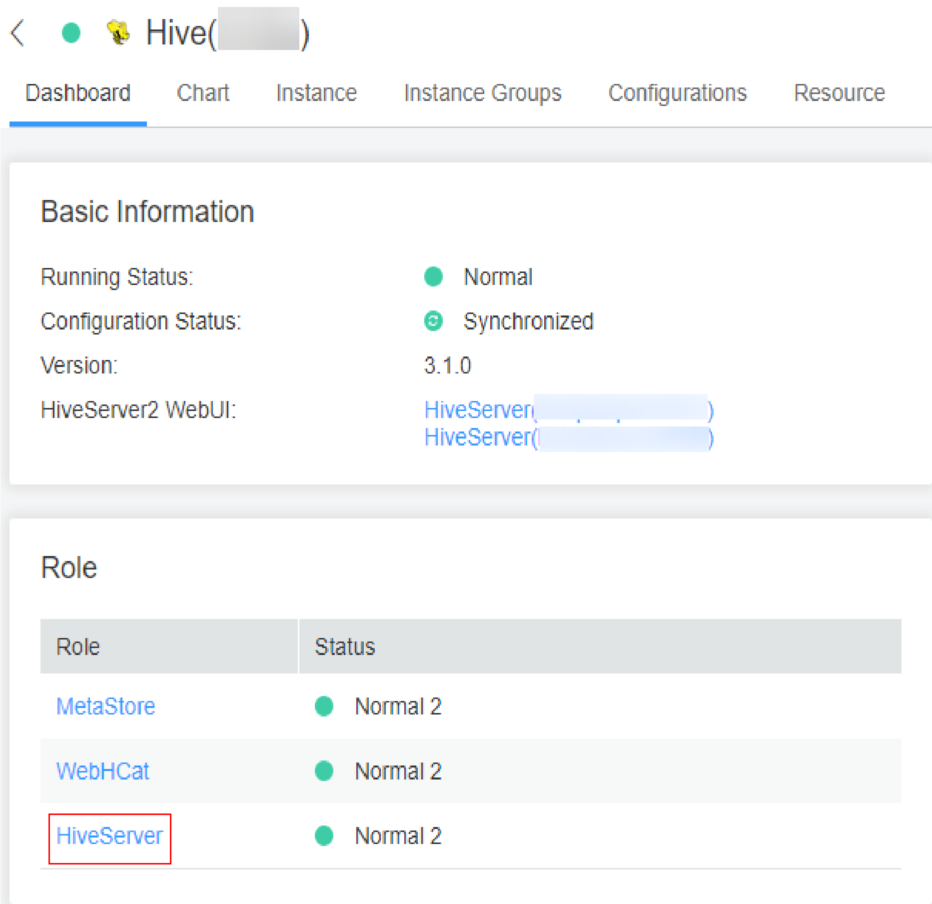
This section applies to MRS 3.*x* or later.

Procedure

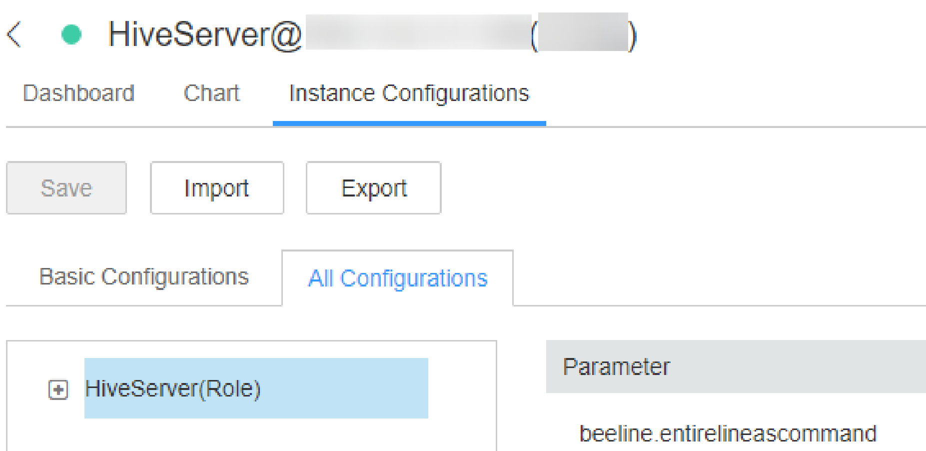
This section describes how to set lease isolation for user **hiveuser** for existing HiveServer instances.

Step 1 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.

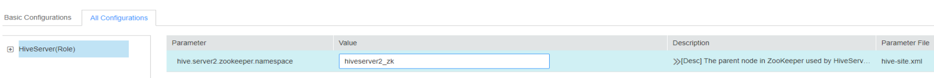
Step 2 Choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **HiveServer**.



Step 3 In the HiveServer list, select the HiveServer for which lease isolation is configured and choose **HiveServer** > **Instance Configurations** > **All Configurations**.



Step 4 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 5 Click **Save**. In the dialog box that is displayed, click **OK**.

Step 6 Choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive**, choose **More** > **Restart Service**, and enter the password to restart the service.

Step 7 Run the **beeline -u** command to log in to the client and run the following command:

beeline -u "jdbc:hive2://*10.5.159.13***:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=***hiveserver2_zk***;sasl.qop=auth-conf;auth=KERBEROS;principal=hive/hadoop.***<System domain name>***@***<System domain name>***"**

In the command, **10.5.159.13** is replaced with the IP address of any ZooKeeper instance, which can be viewed through **Cluster** > *Name of the desired cluster* > **Services** > **ZooKeeper** > **Instance**.

hiveserver2_zk following **zooKeeperNamespace=** is set to the value of **hive.server2.zookeeper.namespace** in **Step 4**.

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

10.7.9 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.

Example

- 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();

10.7.10 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 supports the configuration of the function for creating temporary functions with **ADMIN** permission. The function is disabled by default, which is the same as that of 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 The Hive service configuration page is displayed.

- For versions earlier than MRS 3.x, click the cluster name. On the cluster details page that is displayed, choose **Components** > **Hive** > **Service Configuration**, and select **All** from the **Basic** drop-down list.

  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.)
- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. And choose **Cluster** >

Name of the desired cluster > **Services** > **Hive** > **Configurations** > **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

10.7.11 Allowing Users with Select Permission to View the Table Structure

Scenario

This function is available in Hive and Spark2x in MRS 3.x and later.

With this function enabled, 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 The Hive service configuration page is displayed.

- For versions earlier than MRS 3.x, click the cluster name. On the cluster details page that is displayed, choose **Components** > **Hive** > **Service Configuration**, and select **All** from the **Basic** drop-down list.

  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.)
- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. And choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **Configurations** > **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**. Restart all Hive instances after the modification.

Step 3 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

10.7.12 Allowing Only the Hive Administrator to Create Databases and Tables in the Default Database

Scenario

This function is available in Hive and Spark2x for MRS 3.x or later, or Hive and Spark for versions earlier than MRS 3.x.

After this function is enabled, 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 The Hive service configuration page is displayed.

- For versions earlier than MRS 3.x, click the cluster name. On the cluster details page that is displayed, choose **Components** > **Hive** > **Service Configuration**, and select **All** from the **Basic** drop-down list.

  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.)
- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. And choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **Configurations** > **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**. Restart all Hive instances after the modification.

Step 3 Determine whether to enable this function on the Spark/Spark2x client.

- If yes, go to **Step 4**.
- If no, no further action is required.

Step 4 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. Restart all Spark2x instances after the modification.

Step 5 Download and install the Spark/Spark2x client again.

----End

10.7.13 Configuring Hive to Support More Than 32 Roles

Scenario

This function applies to Hive.

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.
- MRS 3.x and later versions support Ranger. 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 The Hive service configuration page is displayed.

- For versions earlier than MRS 3.x, click the cluster name. On the cluster details page that is displayed, choose **Components** > **Hive** > **Service Configuration**, and select **All** from the **Basic** drop-down list.

  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.)
- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. And choose **Cluster** > *Name of the desired cluster* > **Services** > **Hive** > **Configurations** > **All Configurations**.

Step 2 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**.

Step 3 Choose **HiveServer(Role)** > **Customization**, add a custom parameter to the **hive-site.xml** parameter file, set **Name** to **hive.supports.over.32.roles**, and set **Value** to **true**. Restart all Hive instances after the modification.

----End

10.7.14 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

10.7.15 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.

NOTE

This section applies to MRS 3.*x* or later.

Example

1. Beeline is reconnected after being disconnection.

   Example:

   beeline -e "${*SQL*}" --hivevar batchid=*xxxxx*
2. Beeline kills the running tasks.

   Example:

   beeline -e "" --hivevar batchid=*xxxxx* --hivevar kill=true
3. Log in to the beeline client and start the mechanism of reconnection after disconnection.

   Log in to the beeline client and run the **set hivevar:batchid=***xxxx* command.

NOTE

Instructions:

- *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**.

10.8 Hive Performance Tuning

10.8.1 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 For versions earlier than MRS 3.x:

Log in to the MRS console. In the left navigation pane, choose **Active Clusters** and click a cluster name. Choose **Nodes** > **Node**. The ECS page is displayed. Click **Remote Login** to log in to the Hive node.

For MRS 3.x and later:

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 client tool:

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

10.8.2 Hive Join Optimization

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.

10.8.3 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, and then Reduce 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 clause 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.

10.8.4 Optimizing Hive ORC 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.

10.8.5 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'
      ))
```

10.8.6 Optimizing Query Performance 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

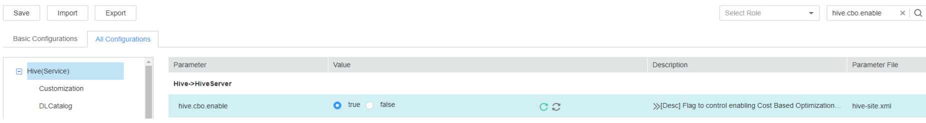
- The CBO optimizes the joining sequence based on statistics and search criteria. However, the joining sequence may fail to be optimized in some special scenarios, such as data skew occurs and query condition values are not in the table.
- When column statistics collection is enabled, Reduce operations must be performed for aggregation. For insert tasks without the Reduce phase, Reduce operations will be performed to collect statistics.
- This section applies to MRS 3.*x* or later.

Prerequisites

You have logged in to the Hive client. For details, see **Using the Hive Client**.

Procedure

Step 1 On the Manager UI, search for the **hive.cbo.enable** parameter in the service configuration of the Hive component, and select **true** to enable the function permanently.



Step 2 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 3 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:

  set hive.stats.autogather = true; enables the automatic collection of table/partition-level statistics.

  set hive.stats.column.autogather = true; enables the automatic collection of column-level statistics.

  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 select **true** to enable the collection function permanently.

Step 4 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

10.9 Hive O&M Management

10.9.1 Hive Common Configuration Parameters

Navigation Path

- For versions earlier than MRS 3.x: Log in to the MRS console. In the navigation pane on the left, choose **Active Cluster**. Click the cluster name and choose **Components** > **Hive** > **Service Configuration**. Click the **Basic Configuration** drop-down list and select **All Configurations**.
- For MRS 3.*x* and later versions: Log in to FusionInsight Manager and choose **Cluster** > **Services** > **Hive**. On the displayed page, click the **Configurations** tab and then **All Configurations**.

Parameter Description

Table 10-9 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.	Versions earlier than MRS 3.x: TextFile For MRS 3.x and later: 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
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. **NOTE** This parameter applies to MRS 3.*x* or later.	true

Parameter	Description	Default Value
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. **NOTE** This parameter applies to MRS 3.*x* or later.	true

10.9.2 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 10-10 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.
	/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.

Log Type	Log File Name	Description
	/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 10-11 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 10-11 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:

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 10-12 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)

10.10 Common Hive SQL Syntax

10.10.1 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 10-13 describes the extended Hive statements provided by .

Table 10-13 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.

10.10.2 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 each search. 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.
- This section applies to MRS 3.*x* or later.

Procedure

Step 1 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 2 Specify the delimiter before search.

set hive.textinput.record.delimiter='!@!';

Hive will use '!@!' as the row delimiter.

----End

10.10.3 Syntax of Traditional Relational Databases Supported by Hive

Overview

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

Syntax description:

- 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            |
  +--------------+--+
  ```

10.11 Common Issues About Hive

10.11.1 How Do I Delete All Permanent Functions from HiveServer?

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

10.11.2 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.

10.11.3 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.

10.11.4 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

10.11.5 What Are the Special Characters Not Supported by Hive in Complex Field Names?

Question

If the name of a Hive complex field contains special characters, the table fails to be created.

Answer

Hive does not support special characters in complex field names. Special characters include uppercase letters, lowercase letters, Arabic numerals, Chinese characters, and Portuguese characters.

10.11.6 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

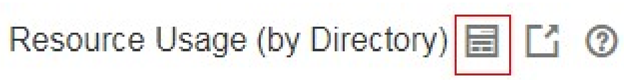
- The Hive and HDFS components are running properly.
- The HDFS refined monitoring function is normal.

Procedure

Step 1 Log in to FusionInsight Manager.

Step 2 Choose **Cluster** > *Name of the desired 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

10.11.7 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

10.11.8 How Do I Handle a Slow Hive on Spark Task 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** > *Name of the desired 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

10.11.9 What Should I Do If an Error Is Reported When the WHERE Condition Is Used to Query Tables with Excessive Partitions in 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.

10.11.10 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

10.11.11 Does the Location of a Hive Table Support Cross-OBS and Cross-HDFS Paths?

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.)

10.11.12 What Should I Do If the MapReduce Engine Cannot Query the Data Written by the Union Statement Running on Tez?

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.

10.11.13 Does Hive 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?

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.

10.11.14 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**.

10.11.15 What Should I Do If the Task Fails When the HDFS Data Directory of the Hive Table Is Deleted By Mistake, But The Metadata Still Exists?

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 an exception caused by misoperation. You need to 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_8.1.0.1/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.

10.11.16 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.

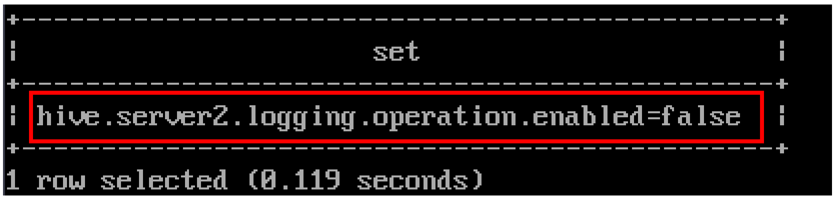
  beeline

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

10.11.17 Why Is the OBS Quick Deletion Directory Not Applied After Being Added to the Custom Hive Configuration?

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.

10.11.18 Hive Configuration Problems

- 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.

10.12 Hive Troubleshooting

10.12.1 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

10.12.2 How Do I Troubleshoot Slow Hive SQL Execution?

Scenario

Possible causes of a Hive SQL task's long running time include slow HiveServer compilation, HDFS access, YARN access, and metadata access.

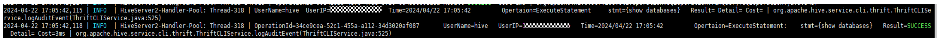
Procedure

To troubleshoot Hive SQL execution slowdown, follow these steps:

- Slow HiveServer Compilation

  View the HiveServer audit log **/var/log/Bigdata/audit/hive/hiveserver/hive-audit.log** and search for the running SQL statement. For example, the SQL statement in **Figure 10-5** is **show databases**. Filter the two records whose thread names contain **HiveServer2-Handler-Pool**. The first record indicates the compilation start time, and the second record indicates the end time. Check the audit records before and after the SQL statement execution becomes slow to determine if the HiveServer compilation is slowing down.

  Figure 10-5 SQL statement running on HiveServer

- Slow HDFS access
  - Method 1:

    View the HiveServer run log file **/var/log/Bigdata/hive/hiveserver/hive.log**, search for the thread log corresponding to the running SQL statement, and then search for the **number of splits** log. If the interval between the logs is long, the HDFS access is slowing down.
  - Method 2:

    Print the **jstack** of the HiveServer process and check whether the related threads are suspended during HDFS access . If yes, the HDFS access is slowing down.
  - Method 3:

    View the HDFS RPC monitoring to check whether the HDFS RPC usage increases abnormally when the SQL statement becomes slow. If yes, there is a high probability that the HDFS access is becoming slow.
- Slow Yarn access

  View the HiveServer run log file **/var/log/Bigdata/hive/hiveserver/hive.log**, search for the thread log corresponding to the running SQL statement, and then search for the **Kill Command** log. If the new line generation in the log takes a long time, the access to YARN is slowing down.
- Slow metadata access

  On FusionInsight Manager, choose **Cluster** > **Services** > **Hive** > **Instances** > *Any MetaStore Instance* > **Chart**. Select **Operations** from the **Chart Category** list. View **Metadata Operation Latency of the create_table API** and **Execution of the add_partitions_req API** to check whether the slow SQL execution is caused by slow MetaStore access.

11 Using Hudi

11.1 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

  MOR tables store data in a hybrid format combining columnar-based Parquet and row-based format Avro. 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.

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

NOTE

To view a Hudi table, 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**.

11.2 Creating a Hudi Table Using Spark Shell

Scenario

This section describes capabilities of Hudi using spark-shell. Using the Spark data source, this section describes how to insert and update a Hudi dataset of the default storage mode Copy-on Write (COW) tables based on code snippets. After each write operation, you will be introduced how to read snapshot and incremental data.

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 (MRS 3.x or Later)**.

NOTE

Currently, Hudi is integrated in Spark2x. You only need to download the Spark2x 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 Use **spark-shell --master yarn-client** to import Hudi packages to 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 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 Query the Hudi table.

Register a temporary table and query the 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 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

11.3 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

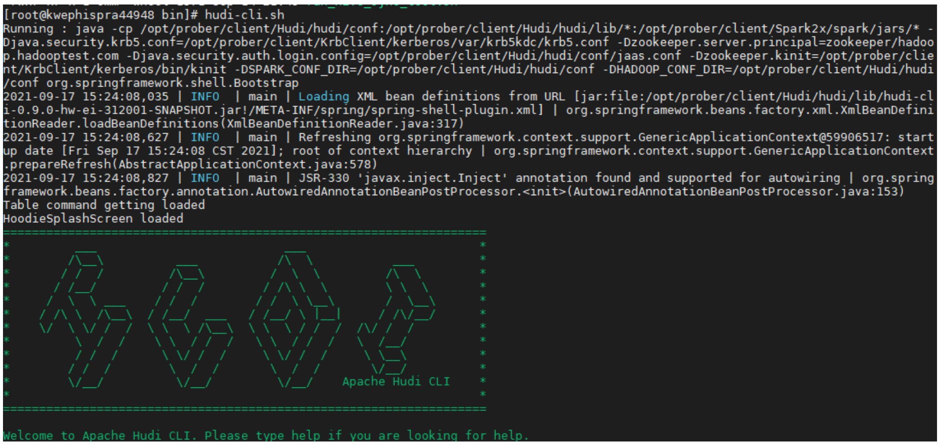
   source Hudi/component_env

   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. In this case, 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.

11.4 Hudi Write Operation

11.4.1 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, duplicate data will exist in the data lake.
- **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. Add the **option("hoodie.datasource.write.operation", "bulk_insert")** parameter to the command for writing data to a Hudi table to set the write mode to bulk_insert. For example:

```
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 11-4**.
- 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 parameter to **operationTime**, same value as **hoodie.datasource.write.partitionpath.field**.
hoodie.datasource.write.keygenerator.class	The default value is **org.apache.hudi.keygen.SimpleKeyGenerator**. You can set it to **org.apache.hudi.keygen.ComplexKeyGenerator**.
hoodie.datasource.hive_sync.partition_extractor_class	Set this item 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.

11.4.2 Writing Data to Hudi Tables in Streams

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**. Different source classes are specified based on different data sources.

--schemaprovider-class com.huawei.bigdata.hudi.examples.DataSchemaProviderExample // Specify the schema required by the Hudi table.

--transformer-class com.huawei.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**.

11.4.3 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 11-1 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
--assume-date-partitioning	Creates partitions in yyyy/mm/dd format to support backward compatibility.	No	false

Command	Description	Mandatory or Not (Yes or No)	Default Value
--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
--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.

11.5 Hudi Read Operation

11.5.1 Read Hudi Data

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 and Hive. For details, see **Table 11-2** and **Table 11-3**.

Table 11-2 COW tables

Query Engine	Real-time View/Read-optimized View	Incremental View
Hive	Y	Y
Spark (SparkSQL)	Y	Y
Spark (SparkDataSource API)	Y	Y

Table 11-3 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

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.

11.5.2 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.

11.5.3 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)
```

11.6 Data Management and Maintenance

11.6.1 Hudi Clustering

Introduction

Clustering reorganizes data layout to improve query performance without affecting the ingestion speed.

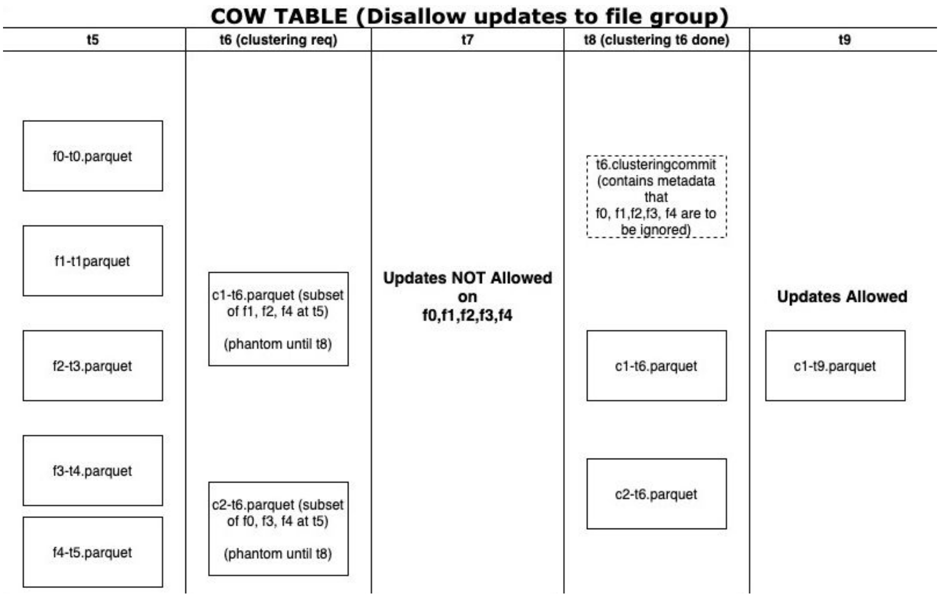
Architecture

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

   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.

11.6.2 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 more cleaning parameters, see **Compaction and Cleaning Configurations**.

11.6.3 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 spark-sql is written, 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").

      ○ · spark-sql can be written by configuring 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.

11.6.4 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_savepoint('hudi_test1', '20220908155421949');
```
- Viewing all existing savepoints
```
call show_savepoints(table =>'hudi_test1');
```
- Rolling back a savepoint
```
call rollback_to_savepoint('hudi_test1', '20220908155421949');
```

NOTE

MOR tables do not support savepoints.

11.7 Typical Hudi Configuration Parameters

11.7.1 Write Configuration

Table 11-4 Write configuration

Parameter	Description	Default Value
hoodie.datasource.write.table.name	Specifies the name of the Hudi table to be written.	None

Parameter	Description	Default Value
hoodie.datasource.write.operation	Specifies the operation type of writing the Hudi table. Currently, **upsert**, **delete**, **insert**, and **bulk_insert** are supported. ● **upsert**: updates and inserts data. ● **delete**: deletes data. ● **insert**: inserts data. ● **bulk_insert**: imports data during initial table creation. Do not **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 combination of **bulk_insert** and **overwrite**.	upsert
hoodie.datasource.write.table.type	Specifies the Hudi table type. Once the table type is specified, this parameter cannot be modified. 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

Parameter	Description	Default Value
hoodie.datasource.write.payload.class	Specifies the class used to merge the records to be updated and the updated records during update. This parameter can be customized. You can compile it yourself to implement your merge logic.	org.apache.hudi.common.model.DefaultHoodieRecordPayload
hoodie.datasource.write.recordkey.field	Specifies the primary key of the Hudi table. The Hudi table must have a unique primary key.	A specific table field
hoodie.datasource.write.partitionpath.field	Specifies the partition key. This parameter is used together with **hoodie.datasource.write.keygenerator.class** to meet the requirements of different partition scenarios.	None
hoodie.datasource.write.hive_style_partitioning	Specifies whether the partition mode is the same as that of Hive. You are advised to set this parameter to **true**.	true
hoodie.datasource.write.keygenerator.class	Generates the primary key and partition mode when used together with **hoodie.datasource.write.partitionpath.field** and **hoodie.datasource.write.recordkey.field**. **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

11.7.2 Configuration of Hive Table Synchronization

Parameter	Description	Default Value
hoodie.datasource.hive_sync.enable	Specifies whether to synchronize the Hudi table information to Hive MetaStore. **CAUTION** You are advised to set this parameter to **true** to use Hive to manage the Hudi table.	false
hoodie.datasource.hive_sync.database	Specifies the name of the database to be synchronized to Hive.	default
hoodie.datasource.hive_sync.table	Specifies the name of the table to be synchronized to Hive. You are advised to set this parameter to the value of **hoodie.datasource.write.table.name**.	unknown
hoodie.datasource.hive_sync.username	Specifies the username used for Hive synchronization.	hive
hoodie.datasource.hive_sync.password	Specifies the password used for Hive synchronization.	hive
hoodie.datasource.hive_sync.jdbcurl	Specifies the Hive JDBC URL for connection.	""
hoodie.datasource.hive_sync.use_jdbc	Specifies whether to use Hive JDBC to connect to Hive for the Hudi table information synchronization. You are advised to set this parameter to **false**, which indicates that the JDBC connection configuration is invalid.	true
hoodie.datasource.hive_sync.partition_fields	Specifies the Hive partition columns.	""
hoodie.datasource.hive_sync.partition_extractor_class	Specifies the class used to extract Hudi partition field values and convert them into Hive partition columns.	org.apache.hudi.hive.SlashEncodedDayPartitionValueExtractor

Parameter	Description	Default Value
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

11.7.3 Index Configuration

Parameter	Description	Default Value
hoodie.index.class	Specifies the full path of user-defined index class. It must be a subclass of HoodieIndex and takes precedence over the **hoodie.index.type** configuration if specified.	""
hoodie.index.type	Specifies the type of index to be used. The Bloom filter is used by default. The possible option is **[BLOOM | HBASE | GLOBAL_BLOOM | SIMPLE | GLOBAL_SIMPLE]**. The Bloom filter eliminates the dependency on an external system and is stored in the footer of a Parquet data file.	BLOOM

Parameter	Description	Default Value
hoodie.index.bloom.num_entries	Specifies the number of entries to be stored in the Bloom filter. If the **maxParquetFileSize** is 128 MB and **averageRecordSize** is 1,024 bytes, a total of 130 KB records exist in a file. The default value (60000) is about half of this approximate value. **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	Specifies the allowed error rate based on 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. Generally, it is set to a small value (0.000000001 by default). You need to balance the disk space to reduce the false positive rate.	0.000000001
hoodie.bloom.index.parallelism	Specifies the parallelism for index lookup, which involves Spark Shuffle. By default, this is automatically computed based on input workload characteristics.	0
hoodie.bloom.index.prune.by.ranges	If this parameter is set to **true**, information is ranged from files to speed up index lookups. It is particularly useful if the key has a monotonously increasing prefix, such as timestamp.	true
hoodie.bloom.index.use.caching	If this parameter is set to **true**, the input RDD is cached to speed up index lookups by reducing I/O for computing parallelism or affected partitions.	true

Parameter	Description	Default Value
hoodie.bloom.index.use.treebased.filter	If this parameter is set to **true**, the file pruning optimization based on interval tree is enabled. This mode speeds up file pruning based on key ranges compared with the brute-force mode.	true
hoodie.bloom.index.bucketized.checking	If this parameter is set to **true**, the bucketized Bloom filtering is enabled. This reduces skew in the sort-based Bloom index lookup.	true
hoodie.bloom.index.keys.per.bucket	This parameter is applicable only when **bloomIndexBucketizedChecking** is enabled and the index type is **BLOOM**. This parameter 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 the Bloom filter lookup. A higher value would amortize the fixed cost of reading the Bloom filter to memory.	10000000
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

Parameter	Description	Default Value
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

11.7.4 Storage 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 x 1024 x 1024 bytes
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 x 1024 x 1024 bytes
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

Parameter	Description	Default Value
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
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

11.7.5 Compaction and Cleaning Configurations

Parameter	Description	Default Value
hoodie.clean.automatic	Whether to perform automatic cleanup.	true

Parameter	Description	Default Value
hoodie.cleaner.policy	Cleaning 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	This parameter controls the number of commit instants read in memory as a batch and archived together.	10
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

Parameter	Description	Default Value
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
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

Parameter	Description	Default Value
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	File groups that will be 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
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

Parameter	Description	Default Value
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

11.7.6 Single-Table Concurrency Control Configuration

Parameter	Description	Default Value
hoodie.write.lock.provider	Specifies the 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	Specifies the Hive database.	None
hoodie.write.lock.hivemetastore.table	Specifies the Hive table name.	None
hoodie.write.lock.client.num_retries	Specifies the retry times.	10
hoodie.write.lock.client.wait_time_ms_between_retry	Specifies the retry interval.	10000

Parameter	Description	Default Value
hoodie.write.lock.conflict.resolution.strategy	Specifies the 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 period.	15000
hoodie.write.lock.zookeeper.port	ZooKeeper port number.	None
hoodie.write.lock.zookeeper.url	URL of the ZooKeeper.	None
hoodie.write.lock.zookeeper.session_timeout_ms	Session expiration time of ZooKeeper.	60000

11.8 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. For details, see **Spark Core Performance Tuning**.

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.

11.9 Common Issues About Hudi

11.9.1 Data Write

11.9.1.1 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.

11.9.1.2 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.

11.9.1.3 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.

11.9.1.4 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")**.

11.9.1.5 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.

11.9.2 Data Collection

11.9.2.1 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'**

11.9.2.2 HoodieException Is Reported When Data Is Collected

Question

The following error is reported when data is collected:

```
com.uber.hoodie.exception.HoodieException: created_at(Part -created_at) field not found in record. Acceptable fields were :[col1, col2, col3, id, name, dob, created_at, updated_at]
```

Answer

This error usually occurs when a field marked as recordKey or partitionKey is not present in the input record. Cross verify the input record.

11.9.2.3 HoodieKeyException Is Reported When Data Is Collected

Question

Is it possible to use a nullable field that contains null records as a primary key when creating a Hudi table?

Answer

No. HoodieKeyException will be thrown.

```
Caused by: org.apache.hudi.exception.HoodieKeyException: recordKey value: "null" for field: "name" cannot
be null or empty.
at org.apache.hudi.keygen.SimpleKeyGenerator.getKey(SimpleKeyGenerator.java:58)
at org.apache.hudi.HoodieSparkSqlWriter$$anonfun$1.apply(HoodieSparkSqlWriter.scala:104)
at org.apache.hudi.HoodieSparkSqlWriter$$anonfun$1.apply(HoodieSparkSqlWriter.scala:100)
```

11.9.3 Hive Synchronization

11.9.3.1 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.in compatible.col.type.changes** to **false**.

11.9.3.2 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.

11.9.3.3 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.

12 Using Hue (Versions Earlier Than MRS 3.x)

12.1 Accessing the Hue Web UI

Scenario

After Hue is installed in an MRS cluster, users can use Hadoop and Hive 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 example, create a human-machine user named **hueuser**, add the user to user groups **hive** (the primary group), **hadoop**, and **supergroup**, and grant it the role **manager_view**.

This user is used to log in to the Hue WebUI.

Procedure

Step 1 Log in to the service page, click the cluster name to go to the cluster details page, and choose **Components**.

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 Select **Hue**. On the right side of **Hue WebUI**, click the link to log in to the Hue web UI as user **hueuser**.

Hue WebUI provides the following functions:

- If Hive is installed in the MRS cluster, you can use **Query Editors** to execute query statements of Hive. Hive has been installed in the MRS cluster.
- If Hive is installed in the MRS cluster, you can use **Data Browsers** to manage Hive tables.
- If HDFS is installed in the MRS cluster, you can use  to view directories and files in HDFS.
- If Yarn is installed in the MRS cluster, you can use  to view all jobs in the MRS cluster.

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 accessing Manager again.

----End

12.2 Using Hue WebUI to Operate Hive Tables

Hue provides the file browser function using a graphical user interface (GUI) so that you can view files and directories on Hive.

Prerequisites

You have installed Hive and Hue, and the Kerberos authentication cluster in the running state.

Procedure

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 Open the Hue web UI and choose **Query Editors** > **Hive**.

Step 3 In **Databases**, select a Hive database, 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 4 Click the desired table name. All columns in the table are displayed.

Step 5 Enter the HiveQL statements in the area for editing.

create table hue_table(id int,name string,company string) row format delimited fields terminated by ',' stored as textfile;

Click and select **Explain**. The editor checks the syntax and execution plan of the entered HiveQL statements. If the statements have syntax errors, the editor reports **Error while compiling statement**.

Step 6 Click , and select the engine for executing the HiveQL statements.

Step 7 Click to execute the HiveQL statements.

Step 8 In the command text box, enter **show tables;** and click . Check whether the **hue-table** table created in **Step 5** exists in the result.

----End

12.3 Using HiveQL Editor on the Hue Web UI

Scenario

Users can use the Hue web UI to execute HiveQL statements in a cluster.

Accessing Query Editors

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 Choose **Query Editors** > **Hive**. The **Hive** page is displayed.

Hive supports the following functions:

- Executes and manages HiveQL statements.
- View the HiveQL statements saved by the current user in **Saved Queries**.
- Query HiveQL statements executed by the current user in **Query History**.
- Click to display all databases included in **Databases** of Hive.

----End

Executing HiveQL Statements

Step 1 Choose **Query Editors > Hive**. The **Hive** page is displayed.

Step 2 Click and select a database from **Databases**. The default database is **default**.

The system displays all available tables in the database. You can enter a keyword of the table name to search for the desired table.

Step 3 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 4 Enter the query statements in the area for editing HiveQL statements.

Click 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 5 Click and select the engine for executing the HiveQL statements.

- **mr**: MapReduce computing framework
- **spark**: Spark computing framework
- **tez**: Tez computing framework

NOTE

Tez is applicable to MRS 1.9.*x* and later versions.

Step 6 Click to execute the HiveQL statements.

NOTE

- If you want to use the entered HiveQL statements again, click to save them.
- To format HiveQL statements, click and select **Format**.
- To delete an entered HiveQL statement, click and select **Clear**.
- Clear the entered statement and execute a new statement. Click and select **New query**.
- Viewing history:

  Click **Query History** to view the HiveQL 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.
- 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 all shortcut keys.

----End

Viewing Execution Results

Step 1 In the **Hive** execution area, **Query History** is displayed by default.

Step 2 Click **Results** to view the execution result of the executed statement.

----End

Managing Query Statements

Step 1 Choose **Query Editors > Hive**. The **Hive** page is displayed.

Step 2 Click **Saved Queries**.

Click a saved statement. The system automatically adds the statement to the editing area.

----End

Modifying Query Editors Settings

Step 1 On the **Hive** tab page, click .

Step 2 Click next to **Files** and click to specify the directory for storing the file.

You can click to add a file resource.

Step 3 Click next to **Functions** and enter the names of user-defined function and function class.

You can click to add a customized function.

Step 4 Click next to **Settings**, enter the Hive parameter name in **Key**, and the value in **Value**. The current Hive session connects to Hive based on the customized configuration.

You can click to add a parameter.

----End

12.4 Using the Metadata Browser on the Hue Web UI

Scenario

Users can use the Hue web UI to manage Hive metadata in an MRS cluster.

Using Metastore Manager

Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Choose **Data Browsers** > **Metastore Tables**, and access **Metastore Manager**.

- Viewing metadata of Hive tables

  In the left navigation pane, move the cursor to a table and click on the right. The metadata of the Hive table is displayed.

- Managing metadata of Hive tables

  On the metadata page of a Hive table, you can click in the upper right corner to import data, click to browse data, and click to view the location of the table file.

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 an operation is required, you are advised to perform the operation on each component after confirming that the operation has no impact on services. For example, you can use the HDFS client to perform operations on HDFS files and use the Hive client to perform operations on Hive tables.

- Managing Hive metadata tables

  Click in the upper right corner to create a table in the database based on the uploaded files. Or click in the upper right corner to manually create a table.

Accessing Metastore Manager

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 Choose **Data Browsers** > **Metastore Tables**, and access **Metastore Manager**.

Metastore Manager supports the following functions:

- Creating a Hive table from a file
- Manually creating a Hive table
- Viewing Hive table metadata

----End

Creating a Hive table from a File

Step 1 Access **Metastore Manager** and select a database in **Databases**.

The default database is **default**.

Step 2 Click . The **Create a new table from a file** page is displayed.

Step 3 Select a file.

1. In **Table Name**, enter a Hive table name.

   A Hive table name contains no more than 128 characters, including letters, numbers, or underscores (_), and must start with a letter or number.
2. In **Description**, enter description about the Hive table as required.
3. In **Input File or Location**, click and select a Hive table file from HDFS. The file is used to store new data of the Hive table.

   If the file is not stored in HDFS, click **Upload a file** to upload the file from the local directory to HDFS. Multiple files can be simultaneously uploaded. The files cannot be empty.
4. If you need to import the data in the file to the Hive table, select **Import data** as **Load method**. By default, **Import data** is selected.

If you select **Create External Table**, a Hive external table is created.

NOTE

If you select **Create External Table**, set **Input File or Location** to a path.

If you select **Leave Empty**, an empty Hive table is created.

5. Click **Next**.

Step 4 Set a delimiter.

1. In **Delimiter**, select one.

   If your desired delimiter is not in the list, select **Other..** and enter a delimiter.

2. Click **Preview** to preview data processing.
3. Click **Next**.

Step 5 Define a column.

1. If you click on the right side of **Use first row as column names**, the first row of data in the file is used as a column name. If you do not click it, the first row of data is not used as the column name.
2. In **Column name**, set a name for each column.

   A Hive table name contains no more than 128 characters, including letters, numbers, or underscores (_), and must start with a letter or number.

   NOTE

   You can rename columns in batches by clicking on the right side of **Bulk edit column names**. Enter all column names and separate them by commas (,).

3. In **Column Type**, select a type for each column.

Step 6 Click **Create Table** to create the table. Wait for Hue to display information about the Hive table.

----End

Manually Creating a Hive Table

Step 1 Access **Metastore Manager** and select a database in **Databases**.

The default database is **default**.

Step 2 Click . The **Create a new table manually** page is displayed.

Step 3 Set a table name.

1. In **Table Name**, enter a Hive table name.

   A Hive table name contains no more than 128 characters, including letters, numbers, or underscores (_), and must start with a letter or number.

2. In **Description**, enter description about the Hive table as required.
3. Click **Next**.

Step 4 Select a data storage format.

- If data needs to be separated by delimiters, select **Delimited** and perform **Step 5**.

- If data needs to be stored in serialization format, select **SerDe** and perform **Step 6**.

Step 5 Set a delimiter.

1. In **Field terminator**, set a column delimiter.

   If your desired delimiter is not in the list, select **Other..** and enter a delimiter.
2. In **Collection terminator**, set a delimiter to separate the data set of columns of the **array** type in Hive. For example, the type of a column is array. A value needs to store **employee** and **manager**. The user specifies a colon (**:**) as the delimiter. Therefore, the final value is **employee:manager**.
3. In **Map key terminator**, set a delimiter to separate the data set of columns of the **map** type in Hive. For example, the type of a column is map. A value needs to store **home** of **aaa** and **company** of **bbb**. The user defines **|** as the delimiter. Therefore, the final value is **home|aaa:company|bbb**.
4. Click **Next** and perform **Step 7**.

Step 6 Set serialization properties.

1. In **SerDe Name**, enter the class name of the serialization format: **org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe**

   Users can expand Hive to support more customized serialization classes.
2. In **Serde properties**, enter the value of the serialization format: **"field.delim"="," "colelction.delim"=":" "mapkey.delim"="|"**
3. Click **Next** and perform **Step 7**.

Step 7 Select a data table format and click **Next**.

- **TextFile**: indicates that data is stored in text files.
- **SequenceFile**: indicates that data is stored in binary files.
- **InputFormat**: indicates that data in files is used in the customized input and output formats.

  Users can expand Hive to support more customized formatting classes.

  a. In **InputFormat Class**, enter the class used by input data: **org.apache.hadoop.hive.ql.io.RCFileInputFormat**

  b. In **OutputFormat Class**, enter the class used by output data: **org.apache.hadoop.hive.ql.io.RCFileOutputFormat**

Step 8 Select a file storage location and click **Next**.

Use default location is selected by default. If you want to customize a storage location, deselect the default value and specify a file storage location in **External location** by clicking .

Step 9 Set columns of the Hive table.

1. In **Column name**, set a column name.

   A Hive table name contains no more than 128 characters, including letters, numbers, or underscores (_), and must start with a letter or number.
2. In **Column type**, select a type for each column.

   Click **Add a column** to add a new column.

3. Click **Add a partition** to add a new partition for the Hive table to improve the query efficiency.

Step 10 Click **Create Table** to create a new table. Wait for Hue to display information about the Hive table.

----End

Managing the Hive Table

Step 1 Access **Metastore Manager** and select a database in **Databases**. All tables in the database are displayed on the page.

The default database is **default**.

Step 2 Click a table name in the database to view table details.

The following operations are supported: importing data, browsing data,, or viewing file storage location. When viewing all tables in the database, you can select tables and perform the following operations such as viewing tables and browsing data.

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 an operation is required, you are advised to perform the operation on each component after confirming that the operation has no impact on services. For example, you can use the HDFS client to perform operations on HDFS files and use the Hive client to perform operations on Hive tables.

----End

12.5 Using File Browser on the Hue Web UI

Scenario

Users can use the Hue web UI to manage files in HDFS in a cluster.

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 an operation is required, you are advised to perform the operation on each component after confirming that the operation has no impact on services. For example, you can use the HDFS client to perform operations on HDFS files and use the Hive client to perform operations on Hive tables.

File Browser (File Browser)

Step 1 Access the Hue web UI.

Step 2 Click . The **File Browser** page is displayed.

You can view the home directory of the current login user.

On the **File Browser** page, the following information about subdirectories for files in the directory is displayed.

Table 12-1 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
Permissions	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

Performing Actions

Step 1 Click  and 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 you can view HDFS storage information about the selected file or directory.

----End

Accessing Other Directories

Step 1 Click the directory name, type a full path you want to access, for example, **/mr-history/tmp**, and press **Enter**.

The current user must have permission to access other directories.

Step 2 Click **Home** to go to the home directory.

Step 3 Click **History**. The history records of directory access are displayed and the directories can be accessed again.

Step 4 Click **Trash** to access the recycle bin of the current directory.

Click **Empty Trash** to clean up the recycle bin.

----End

Uploading User Files

Step 1 Click and click **Upload**.

Step 2 Select an operation.

- **Files**: uploads user files to the current user.
- **Zip/Tgz/Bz2 file**: uploads a compressed file. In the dialog box that is displayed, click **Select ZIP, TGZ or BZ2 files** to select the compressed file to be uploaded. The system automatically decompresses the file in HDFS. Compressed files in **ZIP**, **TGZ**, and **BZ2** formats are supported.

----End

Creating a New File or Directory

Step 1 Click and 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

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.

Step 1 Log in to MRS Manager.

Step 2 On MRS Manager, choose **System** > **Permission > Manage Role** > **Create Role**.

1. Set **Role Name**.
2. Choose **Configure Resource Permission > Hue**, select **Storage Policy Admin**, and click **OK** to grant the storage policy administrator permission to the role.

Step 3 Choose **System** > **Permission** > **Manage User Group** > **Create User Group**, set **Group Name**, and click **Select and Add Role** next to **Role**. On the displayed page, select the created role and click **OK** to add the role to the group.

Step 4 Choose **System** > **Permission** > **Manage User** > **Create User**.

1. Specify the **Username** of a user who can log in to the Hue web UI and has the **Storage Policy Admin** permission.
2. Set **User Type** to **Human-machine**.
3. Set **Password** and **Confirm Password** for logging in to the Hue web UI.
4. Click **Select and Join User Group** next to **User Group**. On the page that is displayed, select the created user group, **supergroup**, **hadoop**, and **hive**, and click **OK**.
5. Set **Primary Group** to **hive**.
6. Click **Select and Add Role** on the right of **Assign Rights by Role**. On the Select snf page that is displayed, select the newly created role and the **System_administrator** role, and click **OK**.
7. Click **OK**. The user is added successfully.

Step 5 Access the Hue web UI.

Step 6 Click  in the upper right corner.

Step 7 Select the check box of the directory and click **Action** on the upper part of the page. Then select **Storage policies**.

Step 8 In the dialog box that is displayed, set a new storage policy and click **OK**.

----End

12.6 Using Job Browser on the Hue Web UI

Scenario

You can use the Hue web UI to query all jobs in the cluster.

Accessing Job Browser

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 Click **Job Browser**.

View the jobs in the cluster.

NOTE

The number on **Job Browser** indicates the total number of jobs in the cluster.

Job Browser displays the following job information.

Table 12-2 MRS job attributes

Attribute	Description
Logs	Log information. If a job has logs, is displayed.
ID	Job ID, which is generated by the system automatically.
Name	Job name
Application Type	Job type
Status	Job status. Possible values are **RUNNING**, **SUCCEEDED**, **FAILED**, and **KILLED**.
User	User who starts the job
Maps	Map progress
Reduces	Reduce progress
Queue	Yarn queue used for job running
Priority	Job running priority
Duration	Job running duration
Submitted	Time when the job is submitted to the MRS cluster

NOTE

If the MRS cluster has Spark, the **Spark-JDBCServer** job is started by default to execute tasks.

----End

Searching for Jobs

Step 1 Enter keywords in **Username** or **Text** on the **Job Browser** page to search for the desired jobs.

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 to open job running logs.

----End

12.7 Typical Hue Configurations

Navigation Path

For details about how to set parameters, see **Modifying Cluster Service Configuration Parameters**.

Parameters

Table 12-3 Hue common parameters

Parameter	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	-

13 Using Hue (MRS 3.*x* or Later)

13.1 Accessing the Hue Web UI

Scenario

After Hue is installed in an MRS cluster, users 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, use a browser (for example, Google Chrome) that is compatible with the Hue web UI. 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

If Kerberos authentication is enabled, the MRS cluster administrator has assigned the permission to use Hive to the user. For details, see **Creating a User** .. For example, create a human-machine user named **hueuser**, add the user to user groups **hive** (the primary group), **hadoop**, and **supergroup**, and grant it the role **manager_view**.

This user is used to log in to Manager.

Procedure

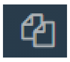
Step 1 Log in to the service page.

For versions earlier than MRS 3.*x*, click the cluster name on the MRS console and choose **Components** > **Hue**.

For MRS 3.*x* or later, log in to FusionInsight Manager and 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, and export 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 accessing Manager again.

----End

13.2 Using Hue WebUI to Operate Hive Tables

Hue aggregates interfaces which interact with most Apache Hadoop components and enables you to use Hadoop components with ease on a web UI. You can operate components such as HDFS, Hive, HBase, Yarn, MapReduce, Oozie, and Spark SQL on the Hue web UI.

Prerequisites

The Hue component has been installed.

Procedure

Step 1 Access the Hue web UI.

Step 2 In the navigation tree on the left, click the editor icon and choose **Hive**.

Step 3 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.

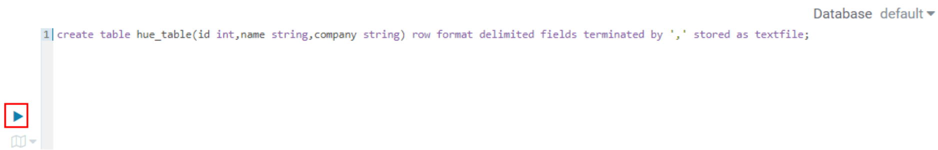
Step 4 Click the desired table name. All columns in the table are displayed.

Step 5 Enter the HiveQL statements in the area for editing.

create table hue_table(id int,name string,company string) row format delimited fields terminated by ',' stored as textfile;

Step 6 Click ▶ to execute the HiveQL statements.

Figure 13-1 Executing a statement



Step 7 In the command text box, enter **show tables;** and click ▶. Check whether the **hue_table** table created in **Step 5** exists in the **Result**.

----End

13.3 Creating a Hue Job

13.3.1 Using HiveQL Editor on the Hue Web UI

Scenario

Users can use the Hue web UI to execute HiveQL statements in an MRS cluster.

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:

- Executes and manages HiveQL statements.
- Views the HiveQL statements saved by the current user in **Saved Queries**.
- Queries HiveQL statements executed by the current user in **Query History**.

----End

Executing HiveQL 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.

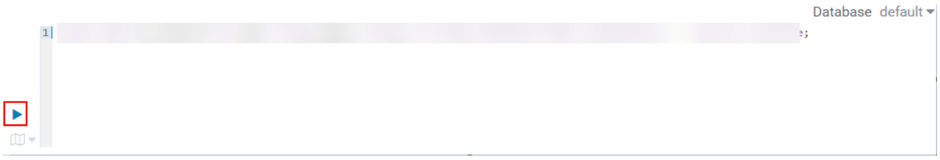
Step 2 Click the desired table name. All columns in the table are displayed.

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 HiveQL statements.

Step 4 Click to execute the HiveQL statements.

Figure 13-2 Executing a statement



NOTE

- If you want to use the entered HiveQL 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 HiveQL statement, click the triangle next to and select **Clear**.
- Viewing history:

  Click **Query History** to view the HiveQL 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

13.3.2 Using the SparkSql Editor on the Hue Web UI

Scenario

You can use Hue to execute SparkSql statements in a cluster on a graphical user interface (GUI).

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 13-1 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**.

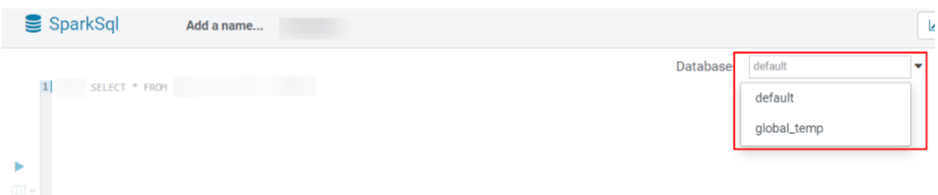
----End

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 13-3 Selecting a database



Step 2 Click the desired table name. All columns in the table are displayed.

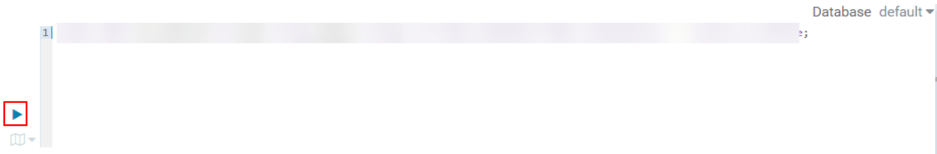
Move the cursor to the row of the table and click . Column details are displayed.

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 13-4 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

13.3.3 Using the Metadata Browser on the Hue Web UI

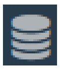
Scenario

Users 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 an operation is required, you are advised to perform the operation on each component after confirming that the operation has no impact on services. For example, you can use the HDFS client to perform operations on HDFS files and use the Hive client to perform operations on Hive tables.

13.3.4 Using File Browser on the Hue Web UI

Scenario

Users can use the Hue web UI to manage files in HDFS in a cluster.

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 an operation is required, you are advised to perform the operation on each component after confirming that the operation has no impact on services. For example, you can 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 13-2 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

Performing Actions

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

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

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

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.

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**.

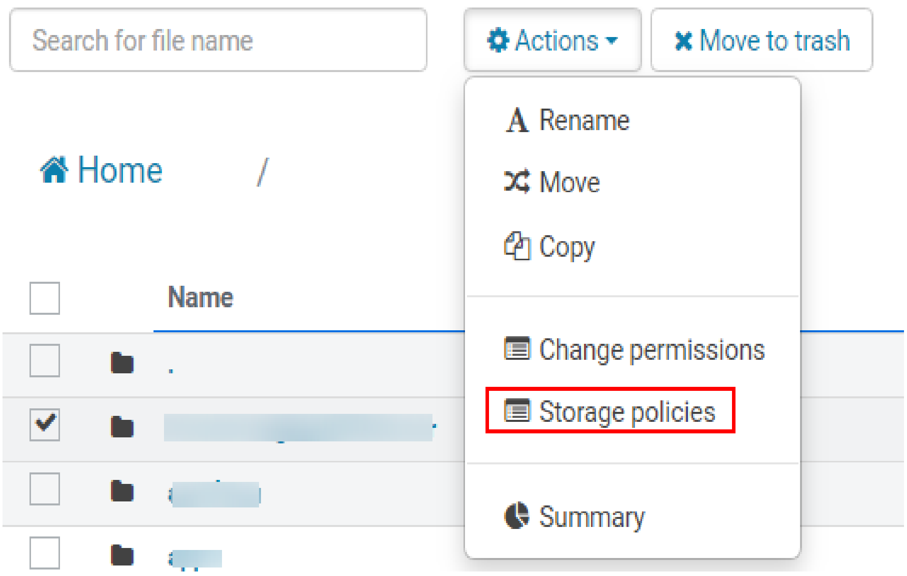
1. **Username**: Enter the name of the user to be added.
2. Set **User Type** to **Human-machine**.
3. Set **Password** and **Confirm Password** for logging in to the Hue web UI.
4. Click **Add** next to **User Group**. On the page that is displayed, select the user group created in **Step 3**, **supergroup**, **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 13-5 Storage policies



Step 8 In the dialog box that is displayed, set a new storage policy and click **OK**.

----End

13.3.5 Using Job Browser on the Hue Web UI

Scenario

Users can use the Hue web UI to query all jobs in an MRS cluster.

Accessing Job Browser

Step 1 Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Step 2 Click .

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 13-3 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

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

13.3.6 Using HBase on the Hue Web UI

Scenario

You can use Hue to create or query HBase tables in a cluster and run tasks on the Hue web UI.

NOTE

If you need to operate HBase on the Hue web UI, the Thrift1Server instance of HBase must be deployed in the MRS cluster.

Thrift1Server instances are not installed by default. When you create a custom MRS cluster, you can select the HBase component and adjust the custom topology of the cluster to add Thrift1Server instances. For details, see **Buying a Custom Topology Cluster**.

If the current cluster supports manual service adding, you can deploy a Thrift1Server instance when you add the HBase service for the first time. After the service is added, restart the Hue service. For details, see **Adding a Service**.

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

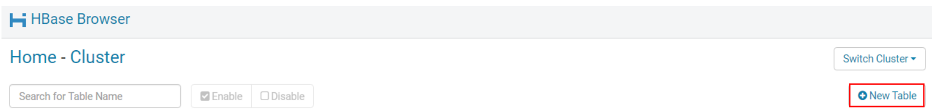
Creating an HBase Table

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 13-6 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.

----End

13.4 Typical Application Scenarios of the Hue Web UI

13.4.1 HDFS on Hue

Hue provides the file browser function for users to use HDFS in GUI mode.

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 an operation is required, you are advised to perform the operation on each component after confirming that the operation has no impact on services. For example, you can use the HDFS client to perform operations on HDFS files and use the Hive client to perform operations on Hive tables.

How to Use File Browser

Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

Click . The **File Browser** page is displayed. You can perform the following operations:

- Viewing files or directories

  By default, the directory and files in the directory of the login user are displayed. You can view **Name**, **Size**, **User**, **Group**,**Permission**, and **Date**.

  Click a file name to view the text information or binary data in the text file. The file content can be edited.

  If there are a large number of files and directories, you can enter keywords in the search text box to search for specific files or directories.

- Creating files or directories

  Click **New** in the upper right corner. Choose **File** to create the file. Choose **Directory** to create a directory.

- Managing files or directories

  Select the check box of a file or director, and click **Actions**. In the displayed menu, choose **Rename**, **Move**, **Copy**, and **Change permissions** to rename, move, copy, or change the file or directory permissions.

- Uploading files

  Click **Upload** in the upper right corner and click **Select files** or drag the file to the window.

How to Use Storage Policies

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

  Current 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, or move the file directory. For details, see **Configuring HDFS Cold and Hot Data Migration**.

  Before configuring a dynamic storage policy on the Hue WebUI, you must set the CRON expressions for cold and hot data migration and start automatic cold and hot data migration on Manager.

  Operations:

  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 13-4 describes the expression for modifying the **dfs.auto.data.mover.cron.expression** parameter. * indicates consecutive time segments.

Table 13-4 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. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.

Step 2 On FusionInsight Manager, choose **System** > **Permission > 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 **Add** next to **Role**. On the displayed page, select the role created in **Step 2**, click **OK** to add the role to the group, and click **OK**.

Step 4 Choose **System** > **Permission** > **User** > **Create**.

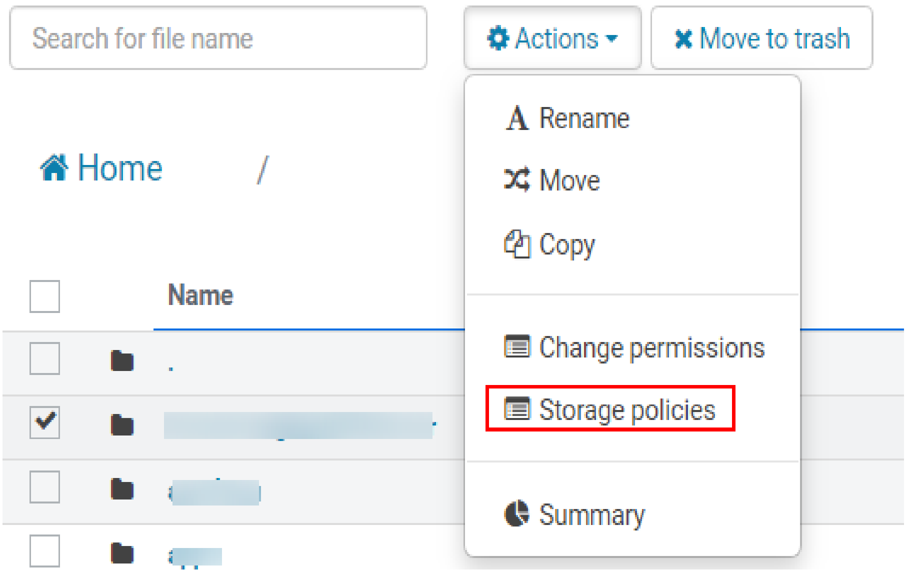
1. **Username**: Enter the name of the user to be added.
2. Set **User Type** to **Human-machine**.
3. Set **Password** and **Confirm Password** for logging in to the Hue web UI.
4. Click **Add** next to **User Group**. On the page that is displayed, select the created user group in **Step 3**, **supergroup**, **hadoop**, and **hive**, and click **OK**.
5. Set **Primary Group** to **hive**.
6. Click **Add** next to **Role**. On the page that is displayed, select the created role in **Step 2** and the **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.

Step 6 In the left navigation pane, click . The **File Browser** page is displayed.

Step 7 Select the check box of a directory and choose **Action** on the top of the page. Choose **Storage policies**.

Figure 13-7 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 13-5** describes the parameters.

Table 13-5 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 Scenarios

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 left navigation pane, click . The **File Browser** page is displayed.

Step 3 Click the name of the file to be viewed.

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

13.4.2 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 13-6 Rule condition tags

Condition Tag	Description
<age operator="lt">	Defines the conditions for changing the age or modification time.
<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 13-7 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 13-8 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.huawei.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 13-9 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.

https://hadoop.apache.org/docs/r3.1.1/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.huawei.hadoop.hdfs.datamovement.HDFSAutoDataMovementTool.audit=$
{autodatatool.logger}
log4j.additivity.com.huawei.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.

13.4.3 Hive on Hue

Hue provides the Hive GUI management function so that users can query Hive data in GUI mode.

How to Use Query Editor

Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

In the navigation tree on the left, click and choose **Hive**. The **Hive** page is displayed.

- Running Hive HQL statements

  Select the target database on the left. You can also click in the upper right corner and enter the target database name to search for the target database.

  Enter a Hive 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 Hive HQL statements

  Select the target database on the left, enter the Hive 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 Hive HQL statement in the text box, click in the upper right corner, and enter the name and description. You can view the saved statements on the **Saved Queries** tab page.

- Viewing historical records

  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.

- Configuring advanced query

  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.

How to Use Metadata Browser

Access 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 an operation is required, you are advised to perform the operation on each component after confirming that the operation has no impact on services. For example, you can use the HDFS client to perform operations on HDFS files and use the Hive client to perform operations on Hive tables.

Typical Scenarios

On the Hue page, create a Hive table as follows:

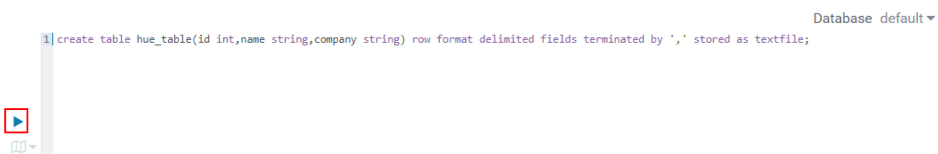
Step 1 Click at the upper left corner of Hue web UI and select the Hive instance to be operated to enter the Hive command execution page.

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 13-8 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**.

----End

13.4.4 Oozie on Hue

Hue provides the Oozie job manager function, in this case, you can use Oozie in GUI mode.

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 an operation is required, you are advised to perform the operation on each component after confirming that the operation has no impact on services. For example, you can use the HDFS client to perform operations on HDFS files and use the Hive client to perform operations on Hive tables.

How to Use Oozie Job Designer

Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

In the navigation tree on the left, click and choose **Workflow**.

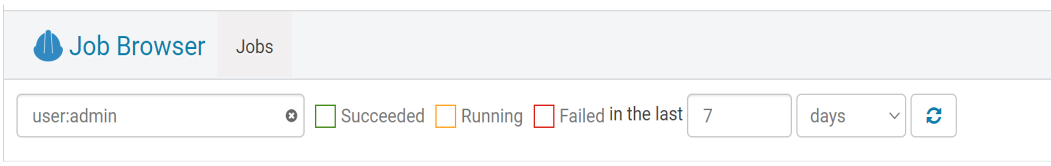
The job designer allows users to create MapReduce, Java, Streaming, Fs, SSH, Shell and DistCp jobs.

How to Use Dashboard

Access the Hue web UI.

Click **Jobs** in the upper right corner. The **Job Browser** page is displayed.

View the running status of the Workflow, Coordinator, and Bundles jobs.



How to Use Editor

Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

In the navigation tree 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.

13.5 Typical Hue Configurations

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 13-10**.

Table 13-10 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 13-11**. The following custom parameters are available only for MRS 3.1.2 or later.

Table 13-11 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**.

13.6 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 13-12 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 13-13 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 13-13 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 13-14 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"}

13.7 Common Issues About Hue

13.7.1 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 the message "There was an error with your query" is displayed?

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.

13.7.2 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?

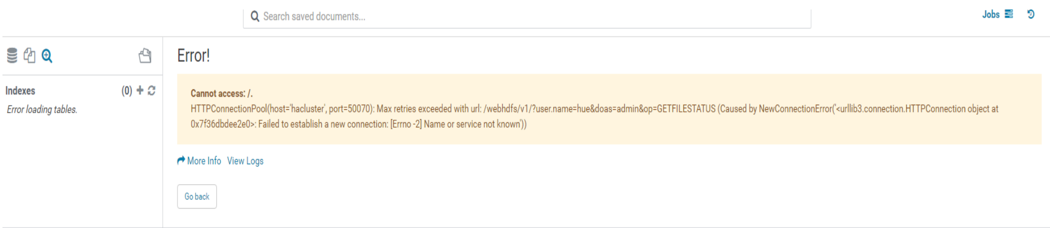
Answer

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.

13.7.3 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.

13.7.4 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.

13.7.5 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, Hue depends on Hive. If this problem occurs, check whether the Hive component is installed in the current cluster. If not, install it.

13.7.6 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.

14 Using Impala

14.1 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 14-1 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 14-1** 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 14-1**. 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

14.2 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.

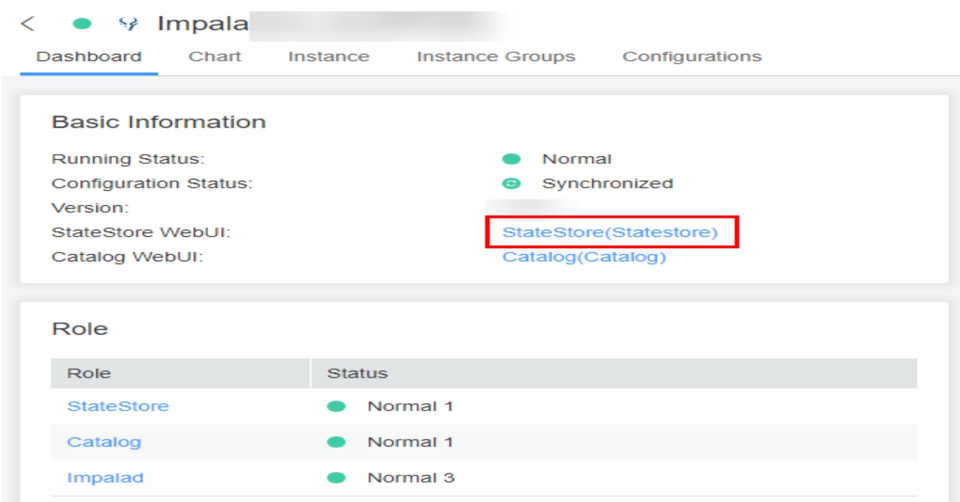
Accessing the StateStore Web UI

Step 1 Access Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.

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 14-1 StateStore WebUI



----End

Accessing the Catalog Web UI

Step 1 Access Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.

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

14.3 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

14.4 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 14-2 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

14.5 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>
```

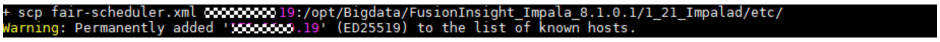


```
<property>
  <name>impala.admission-control.max-query-mem-limit.root.default</name>
  <value>3072000000</value><!--3GB--><!--For reference only-->
</property>
<property>
  <name>impala.admission-control.min-query-mem-limit.root.default</name>
  <value>2048000000</value><!--2GB-->
</property>
<property>
  <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/**

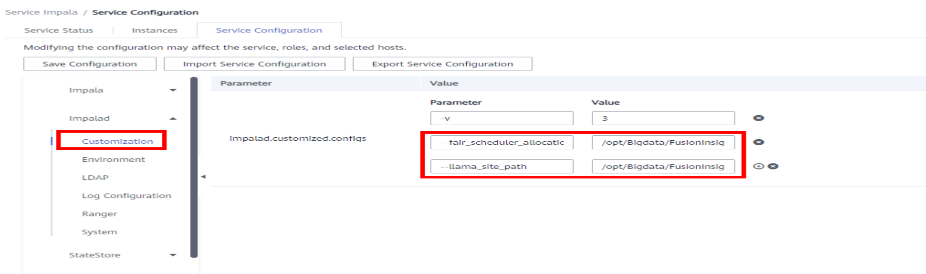
   scp llama-site.xml {IP address of the Impalad instance}:/opt/Bigdata/FusionInsight_Impala_*/***_Impalad/etc/**

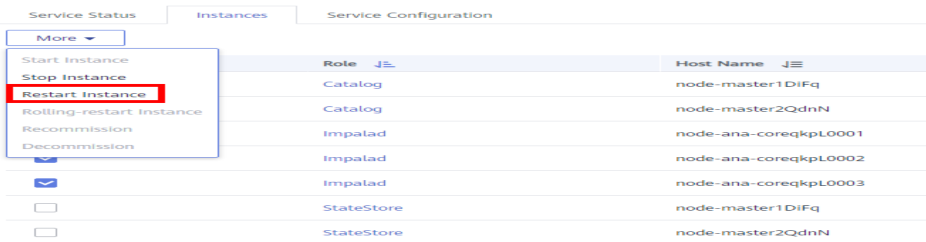
5. Log in to FusionInsight Manager, find the Impala component, and add the following custom configuration items and values to the Impala instance:

   --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**

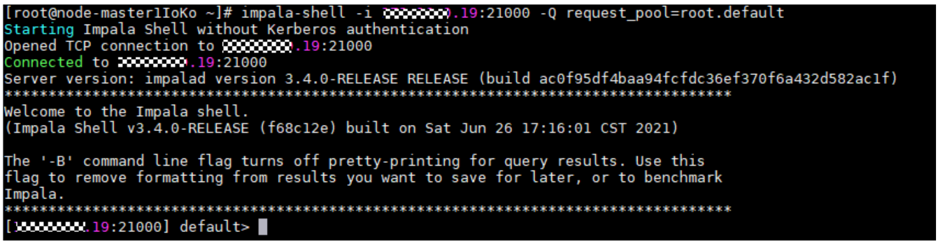
   

6. Restart the Impalad instance.

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:

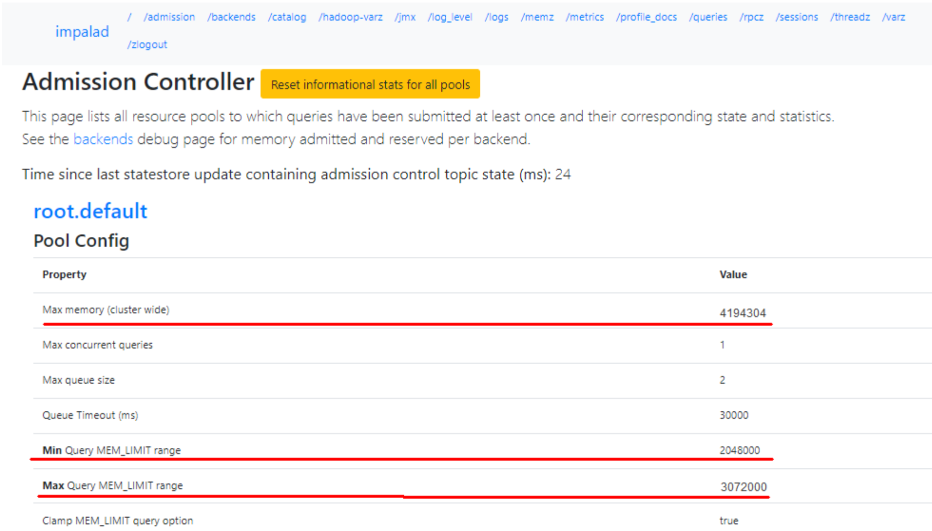
   impala-shell -i {IP address of the Impalad instance:Port number} -Q request_pool=root.default (resource pool configured in fair-scheduler.xml and llama-site.xml)



Execute SQL statements.

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/



14.6 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.

Table 14-3 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**.
--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

14.7 Impala FAQ

14.7.1 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.

15 Using Kafka

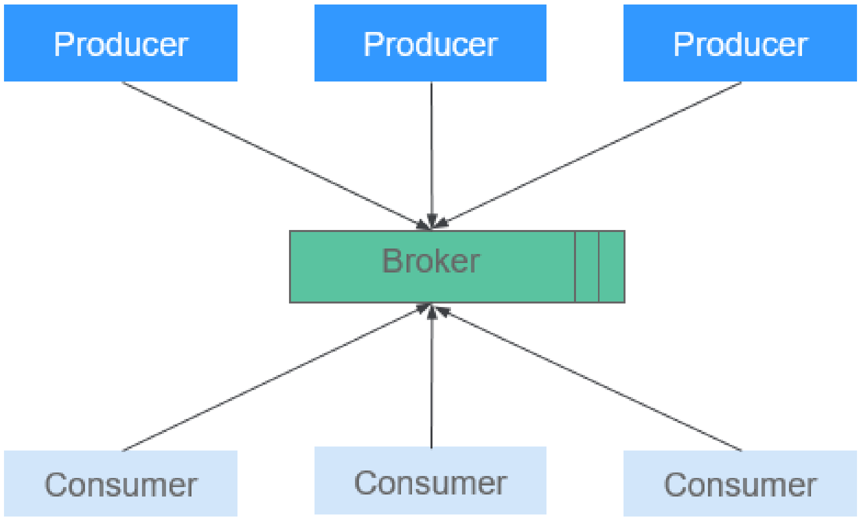
15.1 Kafka Data Consumption

Kafka is an open source, distributed, partitioned, and replicated commit log service. Kafka is publish-subscribe messaging, rethought as a distributed commit log. It provides features similar to Java Message Service (JMS) but another design. It features message endurance, high throughput, distributed methods, multi-client support, and real time. It applies to both online and offline message consumption, such as regular message collection, website activeness tracking, aggregation of statistical system operation data (monitoring data), and log collection. These scenarios engage large amounts of data collection for Internet services.

Kafka Structure

Producers publish data to topics, and consumers subscribe to the topics and consume messages. A broker is a server in a Kafka cluster. For each topic, the Kafka cluster maintains partitions for scalability, parallelism, and fault tolerance. Each partition is an ordered, immutable sequence of messages that is continually appended to - a commit log. Each message in a partition is assigned a sequential ID, which is called offset.

Figure 15-1 Kafka architecture



Kafka UI

Kafka UI provides Kafka web services, displays basic information about functional modules such as brokers, topics, partitions, and consumers in a Kafka cluster, and provides operation entries for common Kafka commands. Kafka UI replaces Kafka Manager to provide secure Kafka web services that comply with security specifications.

You can perform the following operations on Kafka UI:

- Check cluster status (topics, consumers, offsets, partitions, replicas, and nodes).
- Redistribute partitions in the cluster.
- Create a topic with optional topic configurations.
- Delete a topic (supported when **delete.topic.enable** is set to **true** for the Kafka service).
- Add partitions to an existing topic.
- Update configurations for an existing topic.
- Optionally enable JMX polling for broker-level and topic-level metrics.

15.2 Kafka User Permission Management

15.2.1 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 15-1 lists the default Kafka user groups.

NOTE

In MRS 3.*x* or later, 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 15-1 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, for example **admin**, has been prepared.

Procedure

Step 1 Access the ZooKeeper instance page.

- For versions earlier than MRS 3.*x*, click the cluster name to go to the cluster details page, and choose **Components** > **ZooKeeper** > **Instances**.

  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.)

- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > *Name of the desired cluster* > **Services** > **ZooKeeper** > **Instance**.

Step 2 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > *Name of the desired cluster* > **Services** > **ZooKeeper** > **Instance**.

Step 3 View the IP addresses of the ZooKeeper role instance.

Record the IP address of any ZooKeeper instance.

Step 4 Prepare the client based on service requirements. Log in to the node where the client is installed.

Log in to the node where the client is installed. For details, see **Using an MRS Client**. .)

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 Run the following command to authenticate the user(skip this step in normal mode):

kinit *Component service user*

Step 8 Versions earlier than MRS 3.x: Select the scenario required by the service and manage Kafka user permissions.

- Querying the permission list of a topic

  sh kafka-acls.sh --authorizer-properties zookeeper.connect=*IP address of the node where the ZooKeeper instance resides***:2181/kafka --list --topic** *Topic name*

- Adding producer permission to a user

  sh kafka-acls.sh --authorizer-properties zookeeper.connect=*IP address of the node where the ZooKeeper instance resides***:2181/kafka --add --allow-principal User:***Username* **--producer --topic** *Topic name*

- Removing producer permission of a user

  sh kafka-acls.sh --authorizer-properties zookeeper.connect=*IP address of the node where the ZooKeeper instance resides***:2181/kafka --remove --allow-principal User:***Username* **--producer --topic** *Topic name*

- Adding consumer permission to a user

  sh kafka-acls.sh --authorizer-properties zookeeper.connect=*IP address of the node where the ZooKeeper instance resides***:2181/kafka --add --allow-principal User:***Username* **--consumer --topic** *Topic name* **--group** *Consumer group name*

- Removing consumer permission of a user

  sh kafka-acls.sh --authorizer-properties zookeeper.connect=*IP address of the node where the ZooKeeper instance resides***:2181/kafka --remove --allow-principal User:***Username* **--consumer --topic** *Topic name* **--group** *Consumer group name*

NOTE

You need to enter **y** twice to confirm the removal of permission.

Step 9 MRS 3.*x* and later versions: The 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

15.2.2 Creating a Kafka Permission Role

Scenario

Create and configure a Kafka role as an MRS cluster administrator.

This section applies to MRS 3.x or later.

NOTE

Kafka roles can be created in clusters with Kerberos authentication enabled, but not in normal clusters.

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 15-2**.

Table 15-2 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**.
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

15.2.3 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.
- The token authentication mechanism has been enabled. For details, see **Sample Code Running Guide for the Kafka Token Authentication Mechanism**.
- 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

15.3 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.

Prerequisites

- The cluster client has been installed in a directory, for example, **/opt/client**.
- Service users of each component are created by the MRS cluster administrator based on service requirements. A machine-machine user needs to download the **keytab** file and a human-machine user needs to 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 a 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 To perform various operations on Kafka topics for versions prior to MRS 3.*x*, execute the appropriate commands.

- Command for creating 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**

- Command for deleting a topic:

  sh kafka-topics.sh --delete --topic *Topic name* **--zookeeper** *IP address of the node where the ZooKeeper instance resides*:*clientPort*/**kafka**

NOTE

- The number of topic partitions or topic backup copies cannot exceed the number of Kafka instances.
- By default, the ZooKeeper's **clientPort** value is **2181**.
- There are three ZooKeeper instances. Use the IP address of any one.

Step 8 MRS 3.*x* and later versions: Use **kafka-topics.sh** to manage Kafka topics.

- Creating a topic:

  ./kafka-topics.sh --create --topic *Topic name* **--partitions** *Number of partitions occupied by the topic***--replication-factor** *Number of backups of the topic* **--zookeeper** *Service 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 backups of the topic* **--bootstrap-server** *Kafka Cluster IP address*:**21007 --command-config ../config/client.properties**

- Listing topics:
  - **./kafka-topics.sh --list --zookeeper** *Service IP address of any ZooKeeper node*:*clientPort*/**kafka**
  - **./kafka-topics.sh --list --bootstrap-server** *IP address of the Kafka cluster*:**21007 --command-config ../config/client.properties**
- Viewing a 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 Kafka cluster*:**21007 --command-config ../config/client.properties --topic** *Topic name*
- Modifying a topic:
  - **./kafka-topics.sh --alter --topic** *Topic name* **--config** *Configuration item*=*Value***--zookeeper** *Service IP address of any ZooKeeper node*:*clientPort*/**kafka**
- Expanding partitions:
  - **./kafka-topics.sh --alter --topic** *Topic name* **--zookeeper** *Service IP address of any 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 any 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

15.4 Quickly Using Kafka to Produce and Consume Data

Scenario

You can create, query, and delete topics on a cluster client. Set user permissions by referring to **Kafka User Permissions**. Once set, perform operations by referring to **Using the Kafka Client to Produce and Consume Data (for Versions Earlier Than MRS 3.x)**.

For clusters of MRS 3.1.2 or later, you can also log in to the Kafka UI to view the consumption information of the current cluster. For details, see **Using the Kafka UI to View Consumption Information (for MRS 3.1.2 and Later)**.

Prerequisites

- If you plan to use the Kafka client, ensure it is installed in a directory (such as /opt/client). The client directory used in the following operations is just an example and should be replaced with the actual installation directory.
- If you plan to use Kafka UI, ensure you have created a user who has the permission to accessit. 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**.

  Site trust must be added to the browser when you access Manager and Kafka UI for the first time. Otherwise, Kafka UI cannot be accessed.

Using the Kafka Client to Produce and Consume Data (for Versions Earlier Than MRS 3.x)

Step 1 Install the client. For details, see **Installing a Client**.

Step 2 Access the ZooKeeper instance page.

Click the cluster name to go to the cluster details page and choose **Components** > **ZooKeeper** > **Instances**.

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 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 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**

Step 9 Run the following command to view the topic information in the cluster:

sh kafka-topics.sh --list --zookeeper *IP address of the node where the ZooKeeper instance resides:clientPort***/kafka**

Example: **sh kafka-topics.sh --list --zookeeper 10.10.10.100:2181/kafka**

Step 10 Delete the topic created in **Step 8**.

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**

Type **y** and press **Enter**.

----End

Using the Kafka Client to Produce and Consume Data (for MRS 3.x or Later)

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 (MRS 3.x or Later)**. Choose **Cluster** > **Services** > **ZooKeeper** > **Instances**.

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** > **Name of the desired 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**

Step 10 Run the following command to view the topic information in the cluster:

sh kafka-topics.sh --list --zookeeper *IP address of the node where the ZooKeeper instance resides:clientPort***/kafka**

Example: **sh kafka-topics.sh --list --zookeeper 10.10.10.100:2181/kafka**

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 the Kafka UI to View Consumption Information (for MRS 3.1.2 and Later)

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 operations on the page, for example, creating a topic, you need to grant related permissions to users. For details, see **Kafka User Permissions**.

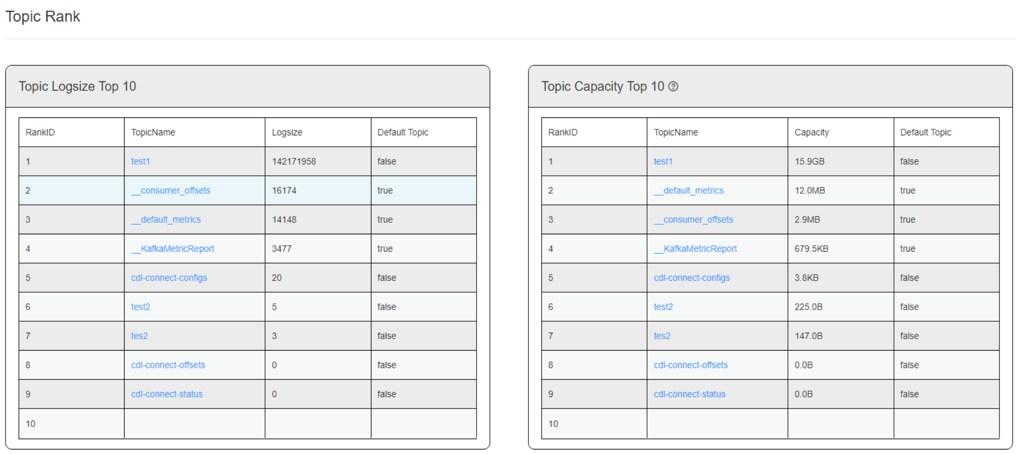
2. On the right of **KafkaManager WebUI**, click the URL to access the Kafka UI.

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, create topics and migrate partitions. For details, see **Adding Kafka Topic Partitions**.

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

15.5 Creating a Kafka Topic

Scenario

You can use the cluster client 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.

- For versions earlier than MRS 3.*x*, click the cluster name to go to the cluster details page, and choose **Components** > **ZooKeeper** > **Instances**.

  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.)

- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > *Name of the desired 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.

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 To perform various operations on Kafka topics for versions prior to MRS 3.*x*, execute the appropriate commands.

- Creating 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**

- Deleting a topic

  sh kafka-topics.sh --delete --topic *Topic name* **--zookeeper** *IP address of the node where the ZooKeeper instance resides:clientPort***/kafka**

NOTE

- The number of topic partitions or topic backup replicas cannot exceed the number of Kafka instances.
- By default, the value of **clientPort** of ZooKeeper is **2181**.
- There are three ZooKeeper instances. Use the IP address of any one.
- For details about managing messages in Kafka topics, see **Managing Messages in Kafka Topics**.

Step 8 MRS 3.*x* and later versions: Use **kafka-topics.sh** to create Kafka topics.

- 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** *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**

- Viewing the 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

15.6 Checking the Consumption Information of 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

1. Ensure that the current consumer is online and consumes data.
2. Configure the **group.id** in the **consumer.properties** configuration file and **--group** in the command to the group to be queried.
3. The Kafka cluster's IP port number is 21007 in security mode and 9092 in normal mode.

----End

Using the Kafka UI to View the Consumption Information (for MRS 3.1.2 and Later)

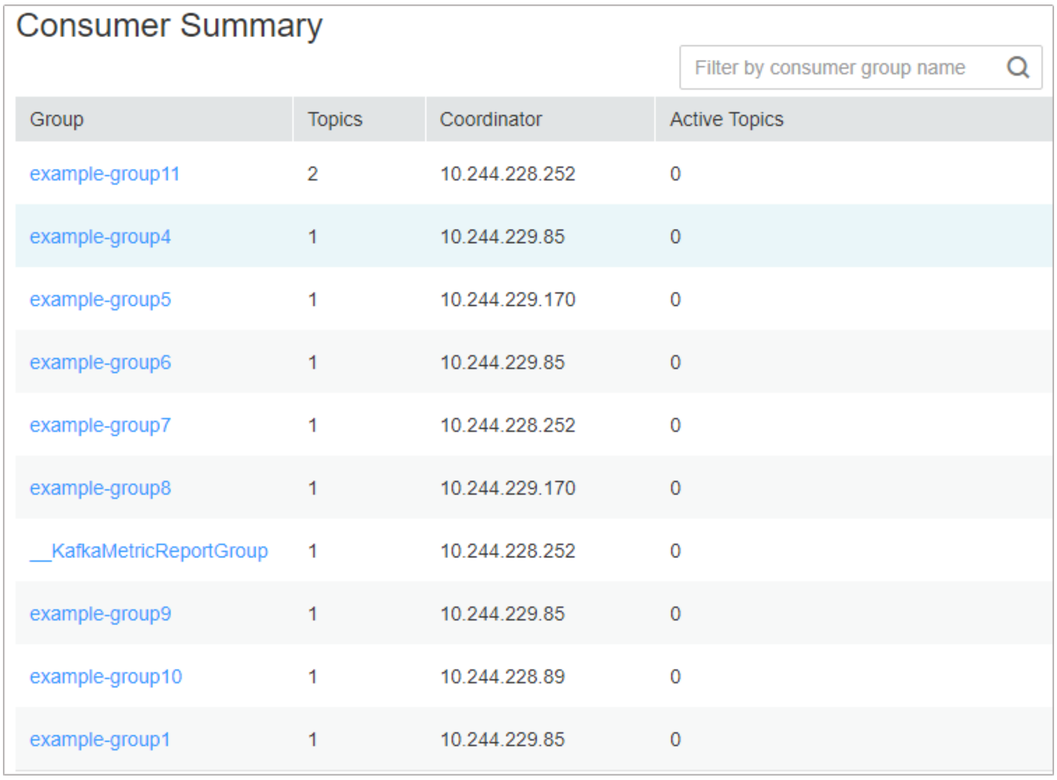
Step 1 Access the Kafka UI.

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 operations on the page, for example, creating a topic, you need to grant related permissions to users. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access the 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.

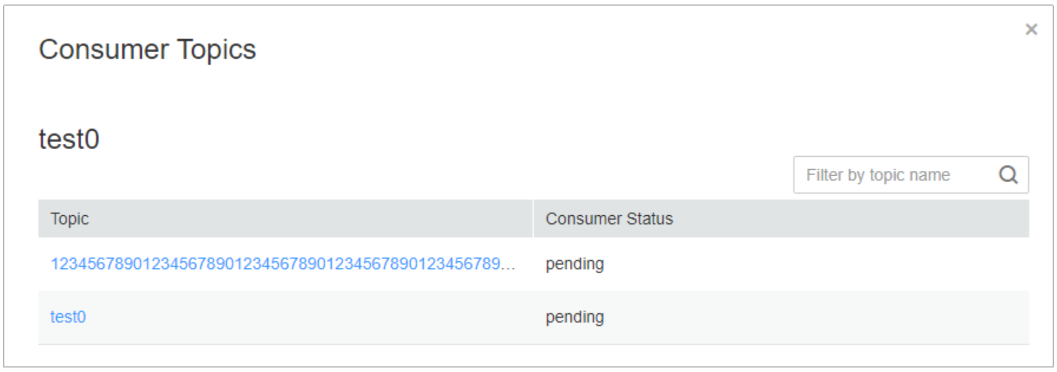


Consumer Summary

Filter by consumer group name

Group	Topics	Coordinator	Active Topics
example-group11	2	10.244.228.252	0
example-group4	1	10.244.229.85	0
example-group5	1	10.244.229.170	0
example-group6	1	10.244.229.85	0
example-group7	1	10.244.228.252	0
example-group8	1	10.244.229.170	0
__KafkaMetricReportGroup	1	10.244.228.252	0
example-group9	1	10.244.229.85	0
example-group10	1	10.244.228.89	0
example-group1	1	10.244.229.85	0

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.

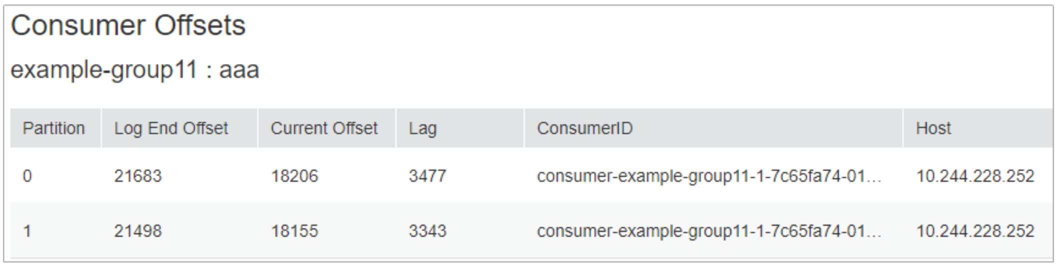

Consumer Topics

test0

Filter by topic name

Topic	Consumer Status
1234567890123456789012345678901234567890123456789...	pending
test0	pending

Step 4 Click a topic name. On the **Consumer Offsets** page that is displayed, view the topic consumption details.

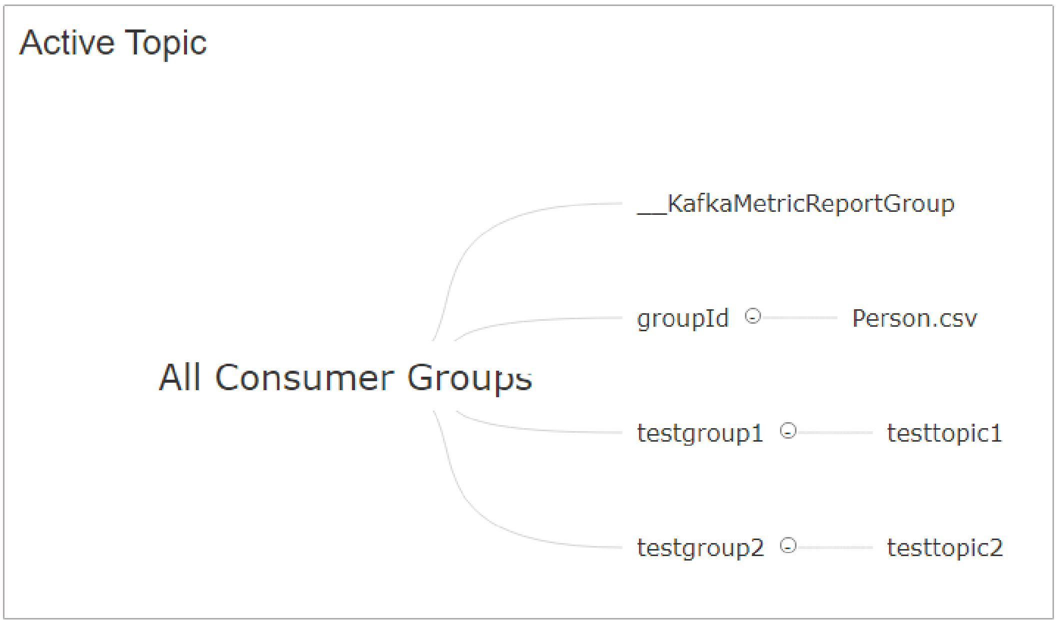

Consumer Offsets

example-group11 : aaa

Partition	Log End Offset	Current Offset	Lag	ConsumerID	Host
0	21683	18206	3477	consumer-example-group11-1-7c65fa74-01...	10.244.228.252
1	21498	18155	3343	consumer-example-group11-1-7c65fa74-01...	10.244.228.252

Step 5 View the consumption relationship diagram.

Click **Consumers**. 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

15.7 Managing Kafka Topics

15.7.1 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 Go to the Kafka service page.

- For versions earlier than MRS 3.x, click the cluster name to go to the cluster details page and choose **Components** > **Kafka**.

  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.)

- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > *Name of the desired 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 the Kafka UI (for MRS 3.1.2 and Later)

Step 1 Access the Kafka UI.

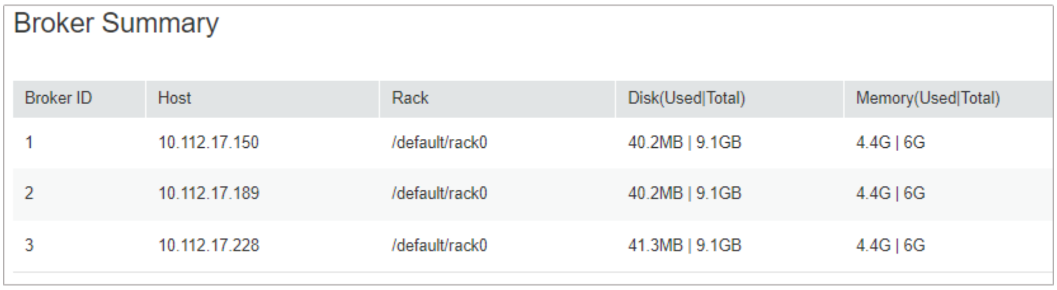
1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

   If you need to perform operations on the page, for example, creating a topic, you need to grant related permissions to users. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access the Kafka UI.

Step 2 Click **Brokers**.

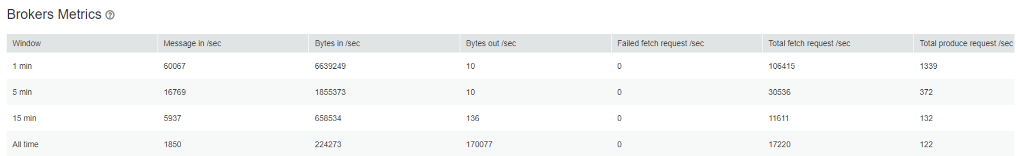
Step 3 In the **Broker Summary** area, you can view **Broker ID**, **Host**, **Rack**, **Disk(Used|Total)**, and **Memory(Used|Total)** of brokers.



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 View the JMX metrics for broker node data traffic in the **Brokers Metrics** area. You can check the average number of incoming messages per second, incoming and outgoing message byte counts per second, failed requests per second, total requests per second, and production requests per second in various 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 Enter a host IP address or rack configuration information in the upper right corner of the page to search for a broker.

----End

15.7.2 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.
- This topic is available for MRS 3.*x* or later.

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 (MRS 3.x or Later)**. 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.

Log in to the node where the client is installed. For details, see **Using an MRS Client**.

Step 4 Switch to the client installation directory, for example, **/opt/client/Kafka/kafka/bin**.

cd /opt/client/Kafka/kafka/bin

Step 5 Configure environment variables.

source /opt/client/bigdata_env

Step 6 Perform user authentication. (Skip this step for normal clusters.)

kinit *Component service user*

Step 7 Use **kafka-topics.sh** to modify the topic.

./kafka-topics.sh --alter --topic *Topic name* **--config** *Configuration item*=*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 Kafka cluster*:**21007 --command-config ../config/client.properties --topic** *Topic name*

----End

Using Kafka UI to Modify Kafka Topics (MRS 3.1.2 and Later Versions)

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 operations on the page, for example, creating a topic, you need to grant related permissions to users. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access the Kafka UI.

Step 2 Choose **Topics**.

Step 3 In the **Operation** column of the item to be modified, choose **Action** > **Config**. On the displayed page, change the values of **Key** and **Value** of the topic. To add multiple items, click +.

Step 4 Click **OK**.

----End

15.7.3 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.
- This section applies only to MRS 3.1.2 or later.

Adding a Partition

Step 1 Access the Kafka UI.

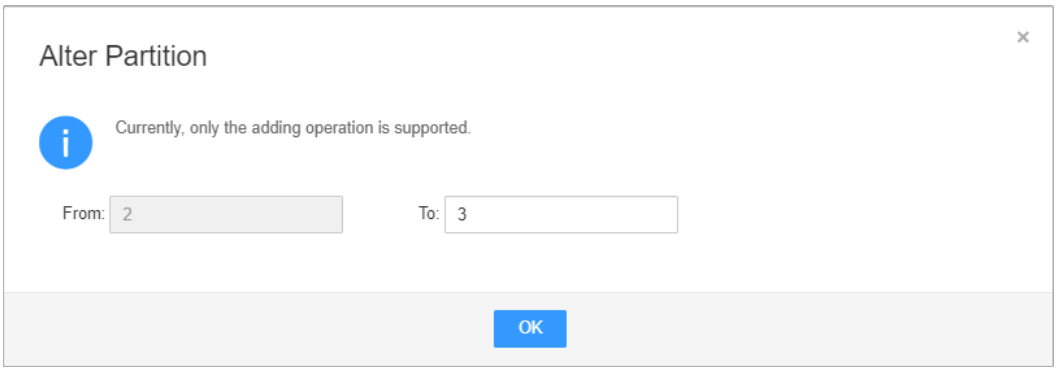
1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

   If you need to perform operations on the page, for example, creating a topic, you need to grant related permissions to users. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access the Kafka UI.

Step 2 Choose **Topics**.

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

15.7.4 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.

Procedure

Step 1 Go to the Kafka service page.

- For versions earlier than MRS 3.*x*, click the cluster name to go to the cluster details page and choose **Components** > **Kafka**.

  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.)

- For MRS 3.*x* or later, log in to FusionInsight Manager and choose **Cluster** > *Name of the desired 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.

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

----End

15.7.5 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.
- This section applies only to MRS 3.1.2 or later.

Viewing Production and Consumption Details

Step 1 Access the Kafka UI.

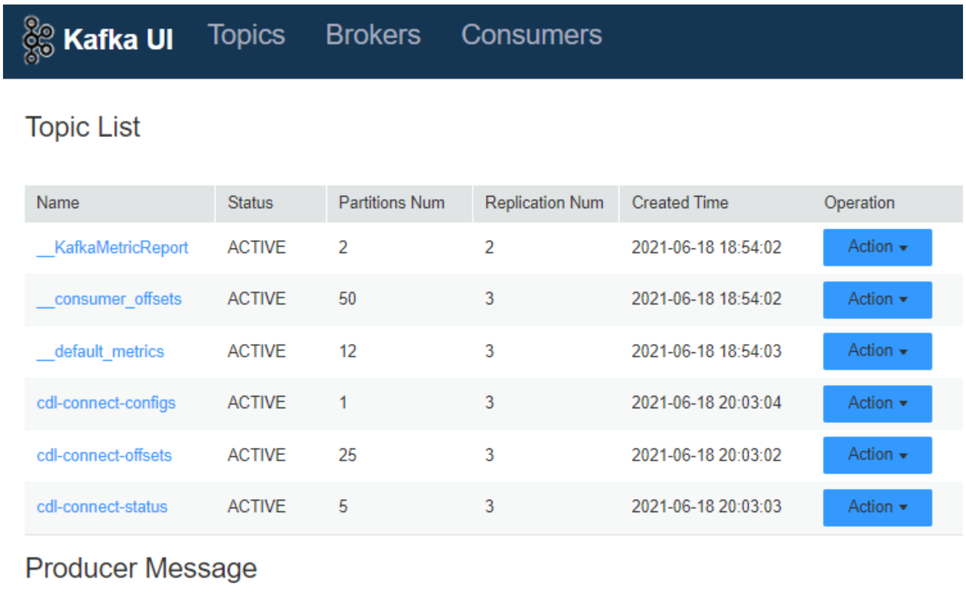
1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

   If you need to perform operations on the page, for example, creating a topic, you need to grant related permissions to users. For details, see **Kafka User Permissions**.

2. On the right of **KafkaManager WebUI**, click the URL to access the 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.



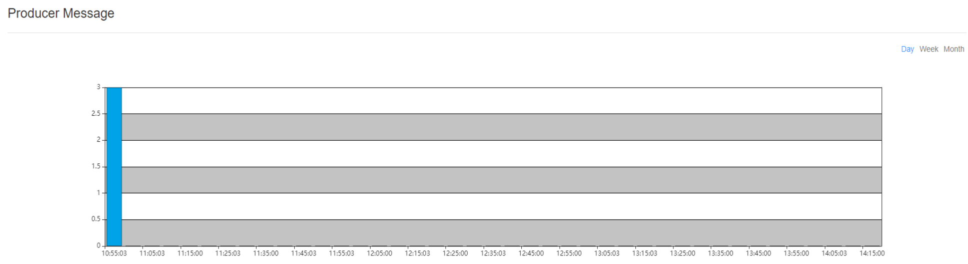
Name	Status	Partitions Num	Replication Num	Created Time	Operation
__KafkaMetricReport	ACTIVE	2	2	2021-06-18 18:54:02	Action
__consumer_offsets	ACTIVE	50	3	2021-06-18 18:54:02	Action
__default_metrics	ACTIVE	12	3	2021-06-18 18:54:03	Action
cdl-connect-configs	ACTIVE	1	3	2021-06-18 20:03:04	Action
cdl-connect-offsets	ACTIVE	25	3	2021-06-18 20:03:02	Action
cdl-connect-status	ACTIVE	5	3	2021-06-18 20:03:03	Action

- 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.

  In the **Operation** column of the item to be modified, choose **Action** > **Config**. On the displayed page, change the values of **Key** and **Value** of the topic. To add multiple items, click +. Click **OK**.
- Search for a topic.

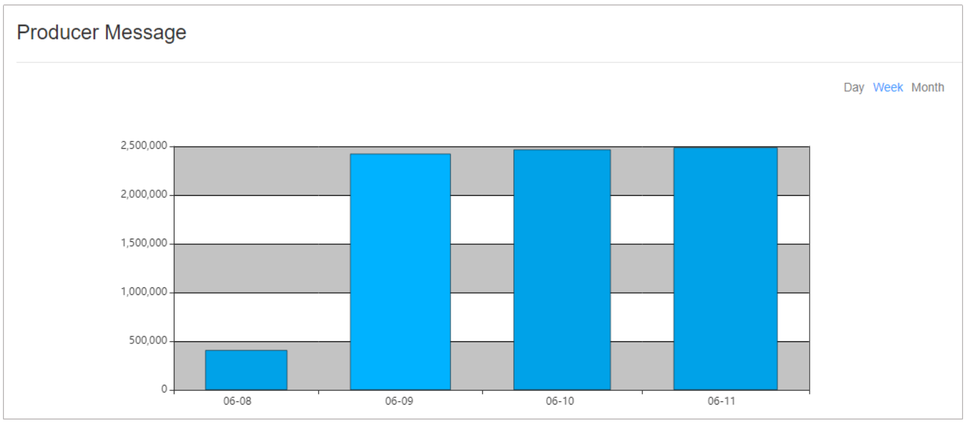
  In the upper right corner of the page, enter a topic name to search for the topic.
- Delete a topic.

  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

15.8 Enterprise-Class Enhancements of Kafka

15.8.1 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:

  NOTICE

  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,

  set the parameters listed in **Table 15-3** on the server. For details about the parameter configuration entry, see **Modifying Cluster Service Configuration Parameters**.

Table 15-3 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 15-4** in the client configuration file **producer.properties**. The path for storing **producer.properties** is **/opt/client/Kafka/kafka/config/producer.properties**, where **/opt/client** indicates the installation directory of the Kafka client.

Table 15-4 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	Default Value	Description
		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 15-5** on the server. For details about the parameter configuration entry, see **Modifying Cluster Service Configuration Parameters**.

Table 15-5 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 15-6** in the client configuration file **producer.properties**. The path for storing **producer.properties** is **/opt/client/Kafka/kafka/config/producer.properties**, where **/opt/client** indicates the installation directory of the Kafka client.

Table 15-6 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-topics.sh --zookeeper 192.168.1.205:2181/kafka --alter --topic test --config unclean.leader.election.enable=false --config min.insync.replicas=2

192.168.1.205 indicates the ZooKeeper service IP address.

- 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.

15.8.2 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

For details about how to view or set parameters, see **Modifying Cluster Service Configuration Parameters**.

The protocols used to access Kafka are as follows: PLAINTEXT, SSL, SASL_PLAINTEXT, and SASL_SSL.

When Kafka service is started, the security authentications using the PLAINTEXT and SASL_PLAINTEXT protocols are started. You can set **ssl.mode.enable** to **true** in Kafka service configuration to start the security authentications using SSL and SASL_SSL protocols. The following table describes the four protocols:

Protocol	Description	Default Port
PLAINTEXT	Supports plaintext access without authentication.	The value of **port**. The default value is 9092.
SASL_PLAINTEXT	Supports plaintext access with Kerberos authentication.	The value of **sasl.port**. The default value is **21007**.
SSL	Supports SSL-encrypted access without authentication.	The value of **ssl.port**. The default value is 9093.

Protocol	Description	Default Port
SASL_SSL	Supports SSL-encrypted access with Kerberos authentication.	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.)
	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.)

Used API	User Group	Client Parameter	Server Parameter	Accessed Port
	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.)
	-	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.)

15.8.3 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.

This section applies to MRS 3.*x* or later. For versions earlier than MRS 3.*x*, see **Balancing Data After Kafka Capacity Expansion**.

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 *<Service IP address of any ZooKeeper node*:*ZooKeeper port*/**kafka**>**--bootstrap-server***<Kafka cluster IP: Port>* **--throttle 10000000 --consumer-config config/consumer.properties --enable-az-aware --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). **--enable-az-aware** indicates that the cross-AZ feature is enabled when the balancing solution is generated. When this parameter is used, ensure that the cross-AZ feature has been enabled for the cluster.

- Run the **--run** command to decommission a node:

  ./bin/kafka-balancer.sh --run --zookeeper *<Service IP address of any ZooKeeper node*:*ZooKeeper port*/**kafka**>**--bootstrap-server***<Kafka cluster IP address: port>* **--throttle 10000000 --consumer-config config/consumer.properties --remove-brokers***<List of broker IDs>* **--enable-az-aware --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. **-enable-az-aware** is optional, indicating that the cross-AZ feature is enabled when the balancing solution is generated. When this parameter is used, ensure that the cross-AZ feature has been enabled for the cluster.

  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:ZooKeeper port/**kafka**>* **--bootstrap-server** *<Kafka cluster IP*

address:port> **--consumer-config config/consumer.properties --enable-az-aware**

This command is used to generate a migration solution based on the current cluster status and print the solution to the console. **--enable-az-aware** is optional, indicating that the cross-AZ feature is enabled when a migration solution is generated. If this parameter is used, ensure that the cross-AZ feature has been enabled for the cluster.

- 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** > *Name of the desired 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, click Cluster, choose **Services** > **Kafka**, and click **Instances** 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.

15.9 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 15-7**.

Table 15-7 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.
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.

15.10 Kafka O&M Management

15.10.1 Kafka Common Configuration Parameters

This section applies to MRS 3.x or later.

Navigation path for setting parameters:

Go to the **All Configurations** page of the Kafka service parameters by referring to **Modifying Cluster Service Configuration Parameters**.

Common Parameters

Table 15-8 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
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 15-9 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	Maximum latency for processing an Offset request.
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	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	Timeout for transferring networkclient connection requests to the Controller and Replica threads on the Broker node. If no response is received within the timeout, the client resends the requests and returns an error message after the maximum number of retries is reached.

Parameter	Description	Default Value	Impact
transaction.max.timeout.ms	Specifies the maximum timeout allowed by the transaction. Unit: millisecond	900,000	Maximum timeout of a transaction. If a request is not responded within the specified timeout, Broker returns an error in InitProducerIdRequest. This prevents a long client request timeout, ensuring that consumer can receive topics.
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 15-10 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 15-11 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.

15.10.2 Kafka Log Overview

This section applies to MRS 3.*x* or later.

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)

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 15-12 Kafka 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

Type	Log File Name	Description
	log-cleaner.log	Cleaner run log of the broker process
	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 15-13 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.

Log Level

Table 15-14 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 15-14 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 15-15 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.

15.10.3 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

15.10.4 Migrating Data on a Kafka Node

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*

```
Topic:testws    PartitionCount:24       ReplicationFactor:2     Configs:
        Topic: testws   Partition: 0    Leader: 4       Replicas: 4,3   Isr: 4,3
        Topic: testws   Partition: 1    Leader: 5       Replicas: 5,4   Isr: 5,4
        Topic: testws   Partition: 2    Leader: 6       Replicas: 6,5   Isr: 6,5
        Topic: testws   Partition: 3    Leader: 3       Replicas: 3,6   Isr: 3,6
        Topic: testws   Partition: 4    Leader: 4       Replicas: 4,5   Isr: 4,5
        Topic: testws   Partition: 5    Leader: 5       Replicas: 5,4   Isr: 5,4
        Topic: testws   Partition: 6    Leader: 6       Replicas: 6,3   Isr: 6,3
        Topic: testws   Partition: 7    Leader: 3       Replicas: 3,4   Isr: 3,4
        Topic: testws   Partition: 8    Leader: 4       Replicas: 4,6   Isr: 4,6
        Topic: testws   Partition: 9    Leader: 5       Replicas: 5,3   Isr: 5,3
        Topic: testws   Partition: 10   Leader: 6       Replicas: 6,4   Isr: 6,4
        Topic: testws   Partition: 11   Leader: 3       Replicas: 3,5   Isr: 3,5
        Topic: testws   Partition: 12   Leader: 4       Replicas: 4,3   Isr: 4,3
        Topic: testws   Partition: 13   Leader: 5       Replicas: 5,4   Isr: 5,4
        Topic: testws   Partition: 14   Leader: 6       Replicas: 6,5   Isr: 6,5
        Topic: testws   Partition: 15   Leader: 3       Replicas: 3,6   Isr: 3,6
        Topic: testws   Partition: 16   Leader: 4       Replicas: 4,5   Isr: 4,5
        Topic: testws   Partition: 17   Leader: 5       Replicas: 5,6   Isr: 5,6
        Topic: testws   Partition: 18   Leader: 6       Replicas: 6,3   Isr: 6,3
        Topic: testws   Partition: 19   Leader: 3       Replicas: 3,4   Isr: 3,4
        Topic: testws   Partition: 20   Leader: 4       Replicas: 4,6   Isr: 4,6
        Topic: testws   Partition: 21   Leader: 5       Replicas: 5,3   Isr: 5,3
        Topic: testws   Partition: 22   Leader: 6       Replicas: 6,4   Isr: 6,4
```

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**. The value of **replicas** must be the same as the number of replicas of the partition. Otherwise, replicas may be missing. In this case, replicas where partitions are located correspond to 6 and 5. If only data in the partition of the node whose **Broker_ID** is **6** is migrated, the data in the partition of the node whose **Broker_ID** is **5** must also be migrated.
- **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 --zookeeper** *{zk_host}:{port}*/**kafka --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 --zookeeper** *{zk_host}:{port}*/**kafka --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 the Kafka UI (MRS 3.1.2 or Later)

Step 1 Access the Kafka UI.

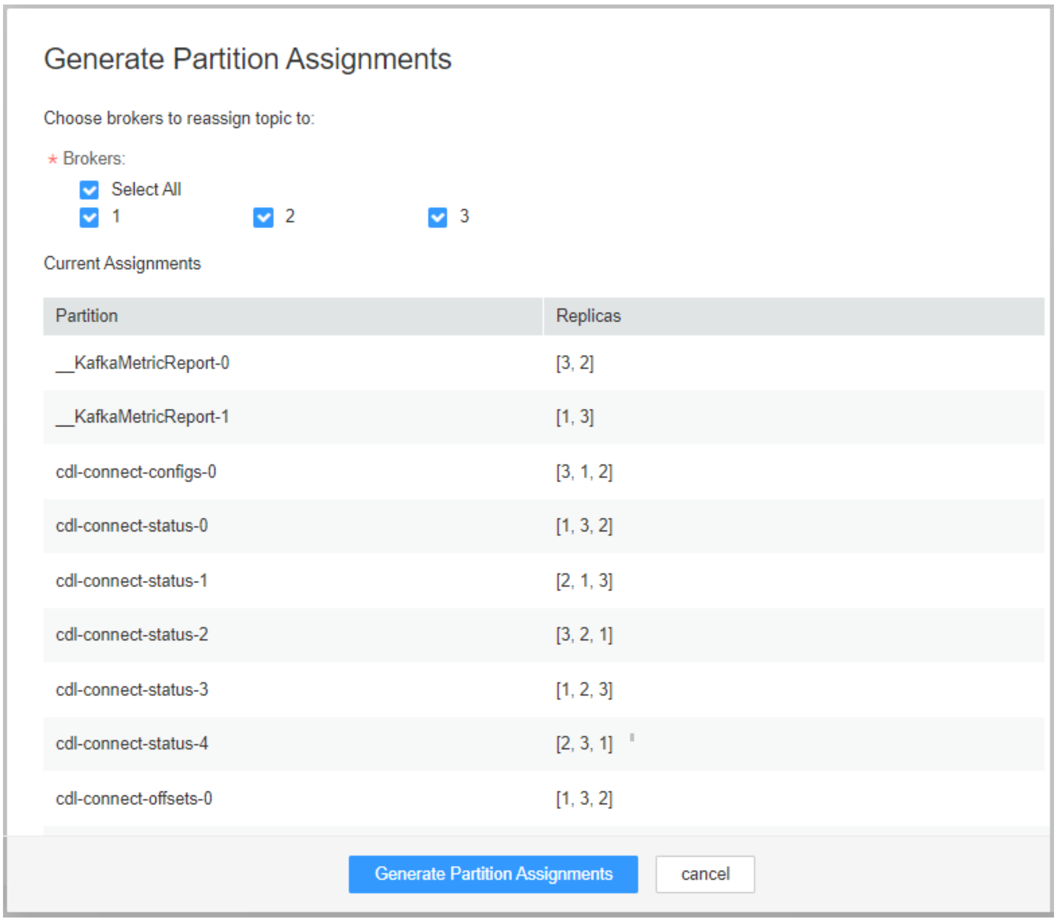
1. Log in to FusionInsight Manager as a user who has the permission to access the Kafka UI and choose **Cluster** > **Services** > **Kafka**.

   If you need to perform operations on the page, for example, creating a topic, you need to grant related permissions to users. For details, see **Kafka User Permissions**.
2. On the right of **KafkaManager WebUI**, click the URL to access the 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

15.10.5 Balancing Data After Kafka Capacity Expansion

Scenario

After adding Kafka nodes, you can run the Kafka balancing tool on the client to distribute the load evenly across the Kafka cluster.

This section applies to versions earlier than MRS 3.*x*. For MRS 3.*x* or later, see **Configuring the Kafka Data Balancing Tool**.

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 in a directory, for example, **/opt/client**.
- Two topics named **test_2** and **test_3** has been created by referring to **Step 7**. The **move-kafka-topic.json** file has been created in the **/opt/client/Kafka/kafka** directory. The topic format is as follows:

```
{
"topics":
[{"topic":"test_2"},{"topic":"test_3"}],
```

```
"version":1
}
```

Procedure

Step 1 Log in to the node where the Kafka client is installed as the client installation user.

Step 2 Run the following command to switch to the client installation directory:

cd /opt/client

Step 3 Run the following command to set environment variables:

source bigdata_env

Step 4 Run the following command to perform user authentication (skip this step if the cluster is in normal mode):

kinit *Component service user*

Step 5 Run the following command to go to the **bin** directory of the Kafka client:

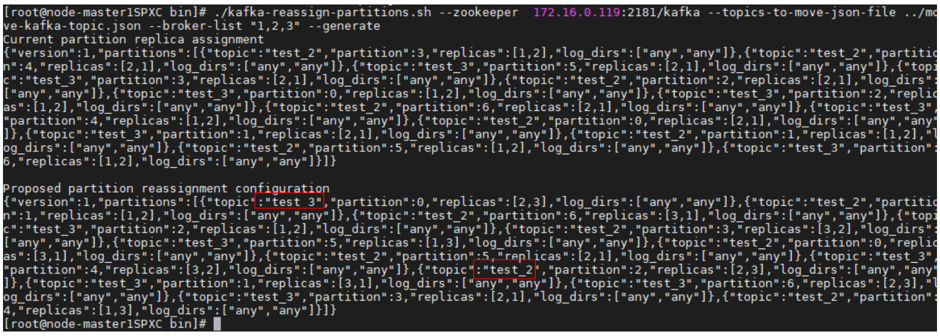
cd Kafka/kafka/bin

Step 6 Run the following command to generate an execution plan:

./kafka-reassign-partitions.sh --zookeeper *172.16.0.119*:**2181/kafka --topics-to-move-json-file ../move-kafka-topic.json --broker-list "***1,2,3***" --generate**

NOTE

- **172.16.0.119**: service IP address of the ZooKeeper instance
- **--broker-list "1,2,3"**: list of broker instances. **1,2,3** indicates all broker IDs after a scale-out.



Step 7 Run the **vim ../reassignment.json** command to create the **reassignment.json** file and save it to the **/opt/kafkaclient/Kafka/kafka** directory.

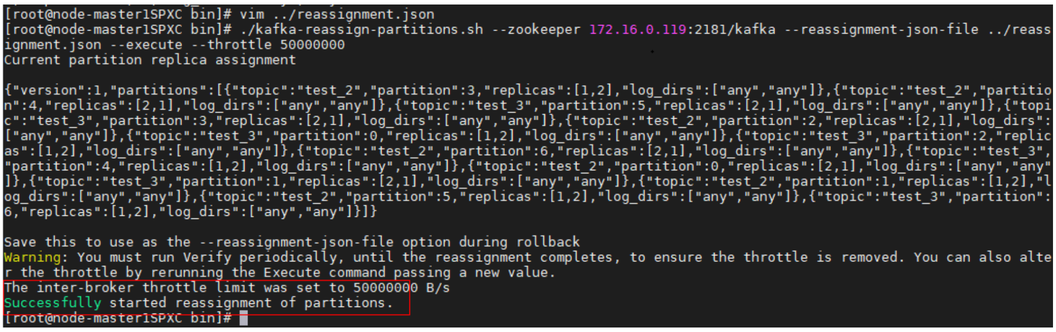
Copy the content under **Proposed partition reassignment configuration** generated in **Step 6** to the **reassignment.json** file, as shown in the follows:

```
{"version":1,"partitions":[{"topic":"test","partition":4,"replicas":[1,2],"log_dirs":["any","any"]},
{"topic":"test","partition":1,"replicas":[1,3],"log_dirs":["any","any"]},{"topic":"test","partition":3,"replicas":
[3,1],"log_dirs":["any","any"]},{"topic":"test","partition":0,"replicas":[3,2],"log_dirs":["any","any"]},
{"topic":"test","partition":2,"replicas":[2,1],"log_dirs":["any","any"]}]}
```

Step 8 Run the following command to redistribute partitions:

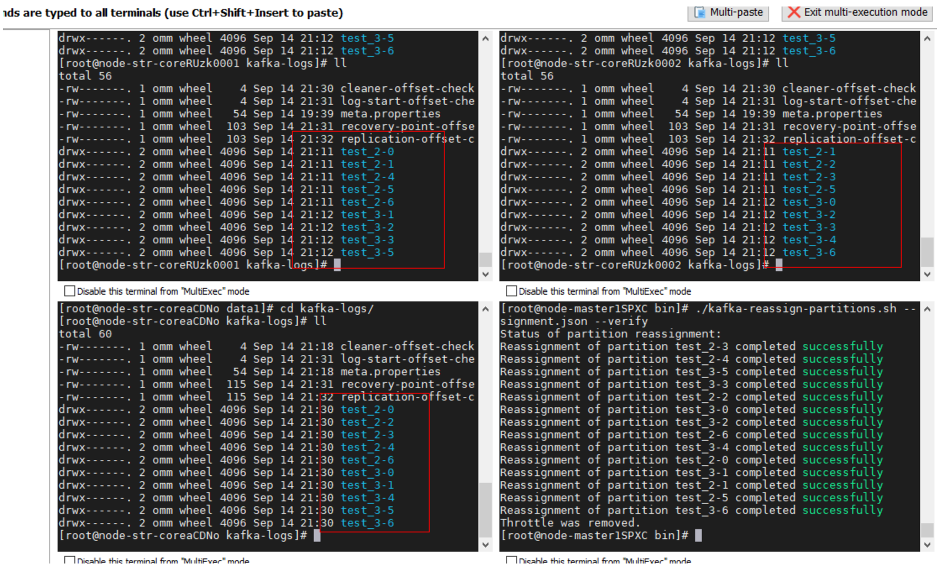
./kafka-reassign-partitions.sh --zookeeper *172.16.0.119*:**2181/kafka --reassignment-json-file ../reassignment.json --execute --throttle** *50000000*

> **NOTE**
>
> **--throttle 50000000**: The maximum bandwidth is 50 MB/s. You can change the bandwidth based on the data volume and the customer's requirements on the balancing time. If the data volume is 5 TB, the bandwidth is 50 MB/s and the data balancing takes about 8 hours.



Step 9 Run the following command to check the data migration status:

./kafka-reassign-partitions.sh --zookeeper *172.16.0.119*:**2181/kafka --reassignment-json-file ../reassignment.json --verify**



----End

15.11 Common Issues About Kafka

15.11.1 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.

15.11.2 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.

15.11.3 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).

- 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 15-16**. If the **conf** directory does not exist, change **config.properties.example** in the **maxwell-XXX** folder to **config.properties**.

Table 15-16 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
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

Parameter	Mandatory	Description	Default Value
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	-
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}

Parameter	Mandatory	Description	Default Value
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.

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.

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 transaction ID.
- **commit**: check whether such operations as data addition, deletion, and modification have been submitted.

15.11.4 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.

16 Using KafkaManager

16.1 Introduction to KafkaManager

KafkaManager is a tool for managing Apache Kafka and provides GUI-based metric monitoring and management of Kafka clusters.

KafkaManager supports the following functions:

- Manage multiple Kafka clusters.
- Check cluster status (topics, consumers, offsets, partitions, replicas, and nodes)
- Run preferred replica election.
- Generate partition assignments with option to select brokers to use.
- Run reassignment of partitions (based on generated assignments).
- Create a topic with optional topic configurations (Multiple Kafka cluster versions are supported).
- Delete a topic (only supported on 0.8.2+ and **delete.topic.enable = true** is set in broker configuration).
- Batch generate partition assignments for multiple topics with option to select brokers to use.
- Batch run reassignment of partitions for multiple topics.
- Add partitions to an existing topic.
- Update configurations for an existing topic.
- Optionally enable JMX polling for broker-level and topic-level metrics.
- Optionally filter out consumers that do not have ids/owner/&offsets/ directories in ZooKeeper.

16.2 Accessing the KafkaManager Web UI

You can monitor and manage Kafka clusters on the graphical KafkaManager web UI.

Prerequisites

- KafkaManager has been installed in a cluster.
- The password of user **admin** has been obtained. The password of user **admin** is specified by the user during MRS cluster creation.

Accessing the KafkaManager Web UI

Step 1 Go to the cluster details page and choose **Components** > **KafkaManager**.

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 **KafkaManager Summary** area, click any UI link in **KafkaManager WebUI** to access the KafkaManager web UI.

You can view the following information on the KafkaManager web UI.

- Kafka cluster list
- Broker node list and metric monitoring information of Kafka clusters
- Kafka cluster replica monitoring information
- Kafka cluster consumer monitoring information

NOTE

You can click the KafkaManager logo in the upper left corner on any sub-page of KafkaManager to return to the homepage of the KafkaManager web UI, where a cluster list is displayed.

----End

16.3 Managing Kafka Clusters

Kafka cluster management includes the following operations:

- **Adding a Cluster on the KafkaManager Web UI**
- **Updating Cluster Parameters**
- **Deleting a Cluster on the KafkaManager Web UI**

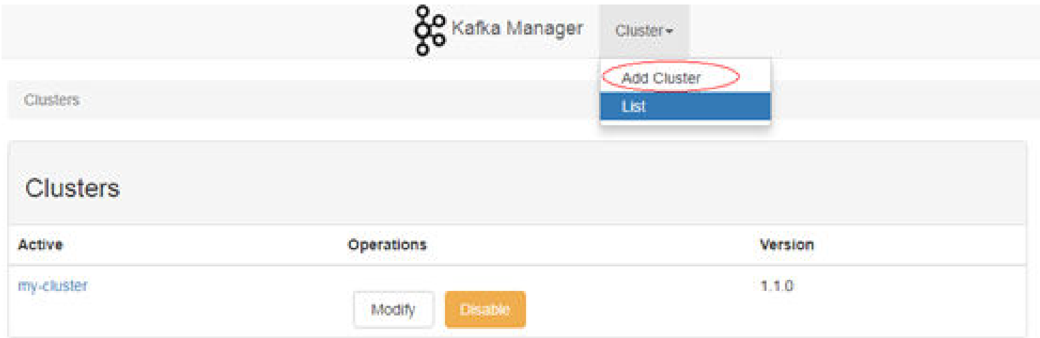
Adding a Cluster on the KafkaManager Web UI

After a Kafka cluster is created for the first time, a default Kafka cluster named **my-cluster** is created on the KafkaManager web UI. You can also add Kafka clusters that have been created on the MRS management console on the KafkaManager web UI to manage multiple Kafka clusters.

Step 1 Log in to the KafkaManager web UI.

Step 2 In the upper part of the page, choose **Cluster** > **Add Cluster**.

Figure 16-1 Adding a cluster



Step 3 Set the cluster parameters. For the following parameters, refer to their example values. Retain the default values for other parameters.

Table 16-1 Cluster parameters to be modified

Parameter	Example Value	Description
Cluster Name	mrs-demo	Name of the cluster to be added on the KafkaManager web UI
Cluster Zookeeper Hosts	zk1_ip:zk1_port, zk2_ip:zk2_port/kafka	ZooKeeper address of the cluster to be added
Kafka Version	1.1.0	Kafka version of the cluster to be added. The default value is **1.1.0**.
Enable JMX Polling (Set JMX_PORT env variable before starting kafka server)	Selected	-
Poll consumer information (Not recommended for large # of consumers)	Selected	-
Enable Active OffsetCache (Not recommended for large # of consumers)	Selected	-
Display Broker and Topic Size (only works after applying this patch)	Selected	-

Parameter	Example Value	Description
Security Protocol	PLAINTEXT	• For a Kafka cluster with Kerberos authentication enabled, select **SASL_PLAINTEXT**. • For a Kafka cluster with Kerberos authentication disabled, select **PLAINTEXT**.

Step 4 Click **Save**.

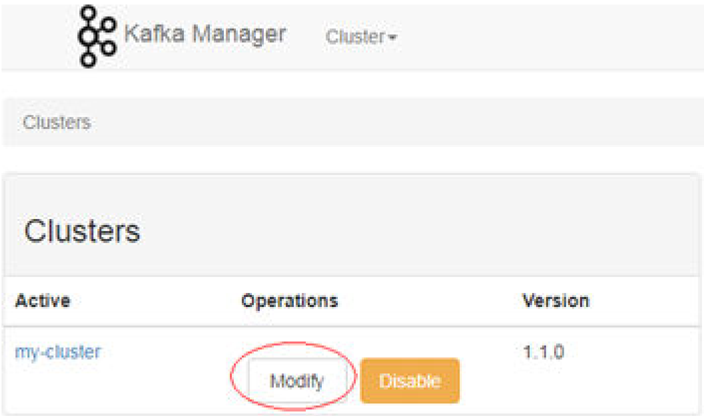
----End

Updating Cluster Parameters

Step 1 Log in to the KafkaManager web UI.

Step 2 Click **Modify** in the **Operations** column of the cluster.

Figure 16-2 Updating cluster parameters



Step 3 Go to the cluster configuration page and modify cluster parameters.

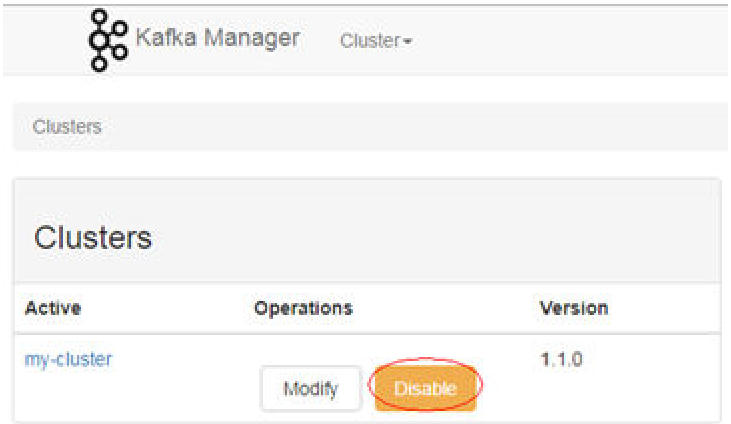
----End

Deleting a Cluster on the KafkaManager Web UI

Step 1 Log in to the KafkaManager web UI.

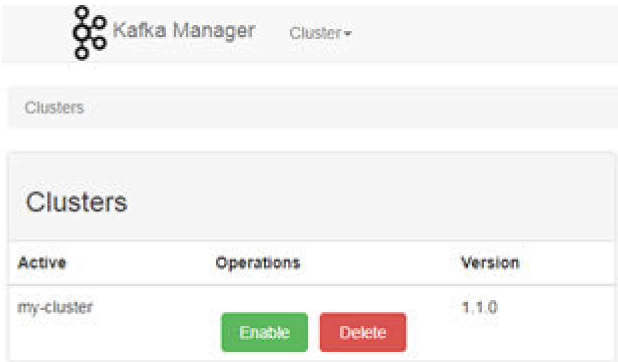
Step 2 Click **Disable** in the **Operations** column of the cluster.

Figure 16-3 Disabling a cluster



Step 3 When **Delete** or **Enable** is displayed in the **Operations** column on the cluster list page, click **Delete** to delete the cluster. You can also click **Enable** to enable the cluster.

Figure 16-4 Enabling or deleting a cluster



----End

16.4 Kafka Cluster Monitoring Management

The Kafka cluster monitoring management includes the following operations:

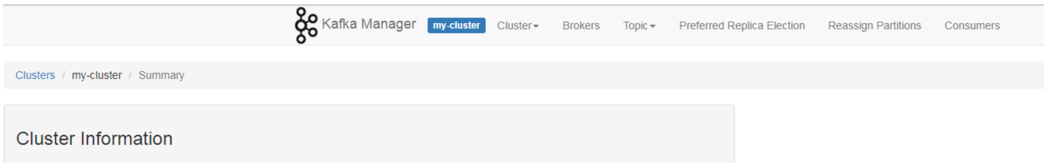
- **Viewing Broker Information**
- **Viewing Topic Information**
- **Viewing Consumers Information**
- **Modifying the Partition of a Topic Through KafkaManager**

Viewing Broker Information

Step 1 Log in to the KafkaManager web UI.

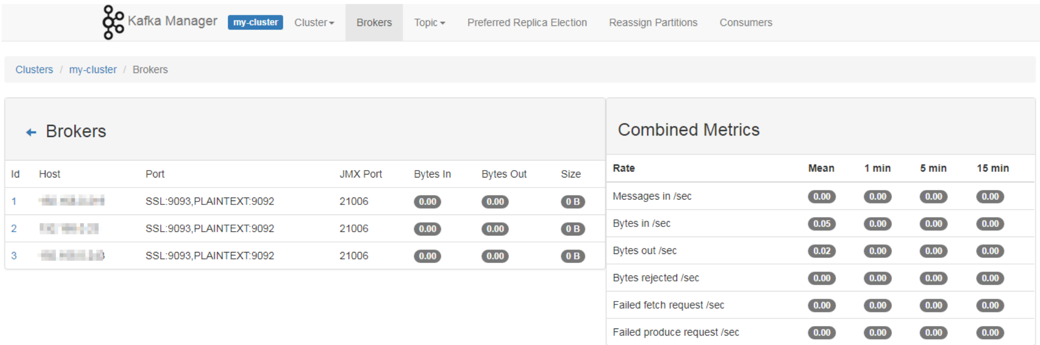
Step 2 On the cluster list page, click a cluster name to access the Summary page of the cluster.

Figure 16-5 Summary page of a cluster



Step 3 Click **Brokers** to access the Broker monitoring page. The page displays the Broker list and I/O statistics of the Broker nodes.

Figure 16-6 Broker monitoring page



----End

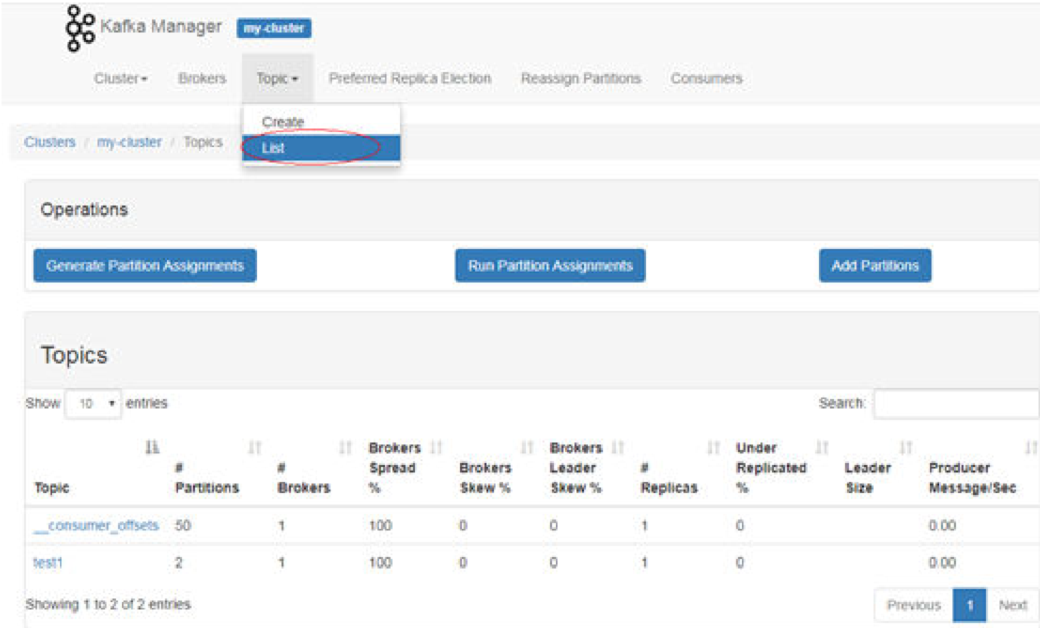
Viewing Topic Information

Step 1 Log in to the KafkaManager web UI.

Step 2 On the cluster list page, click a cluster name to access the **Summary** page of the cluster.

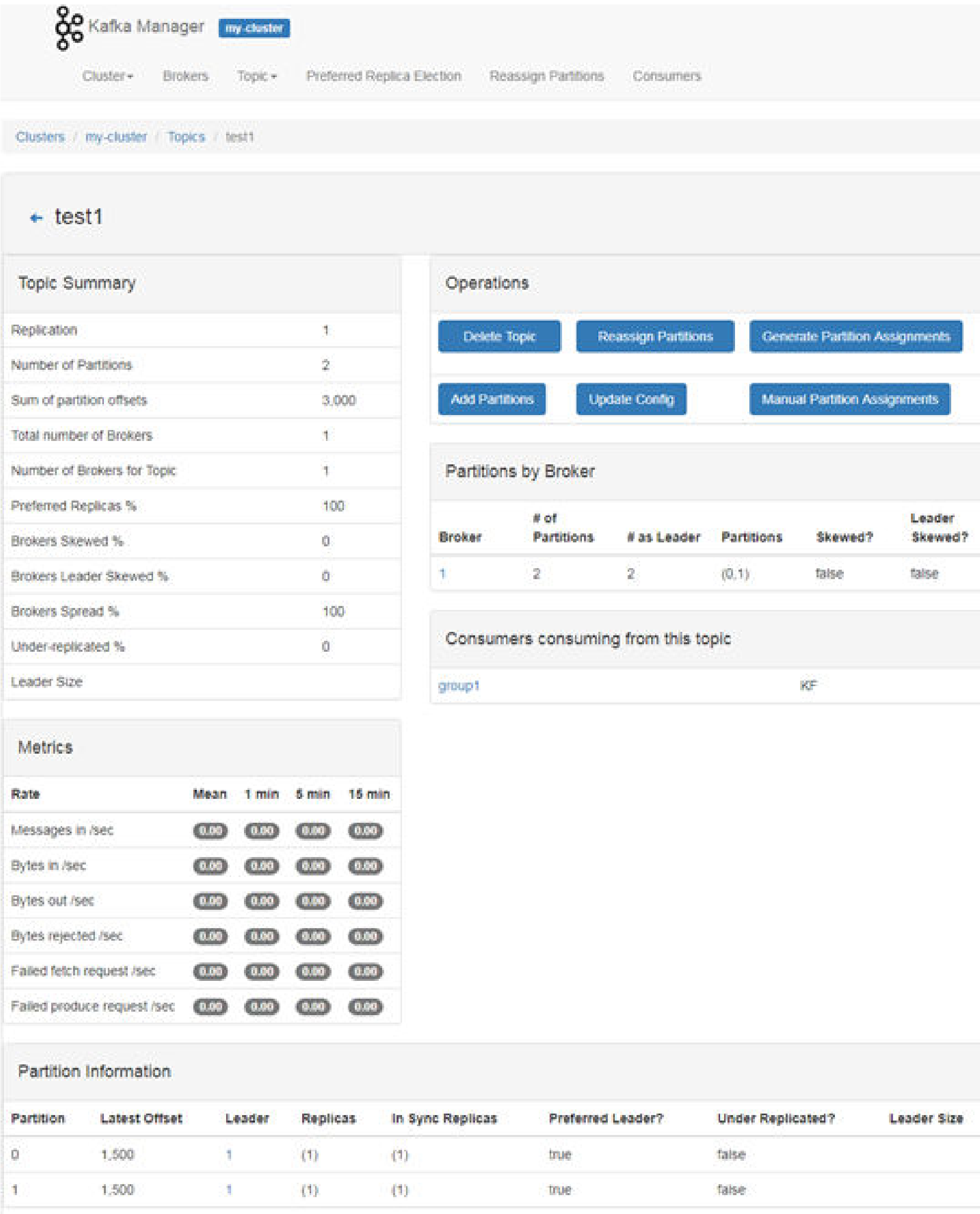
Step 3 Choose **Topic** > **List** to view the topic list of the current cluster and information about each topic.

Figure 16-7 Topic list



Step 4 Click a topic name to view details about the topic.

Figure 16-8 Topic details



----End

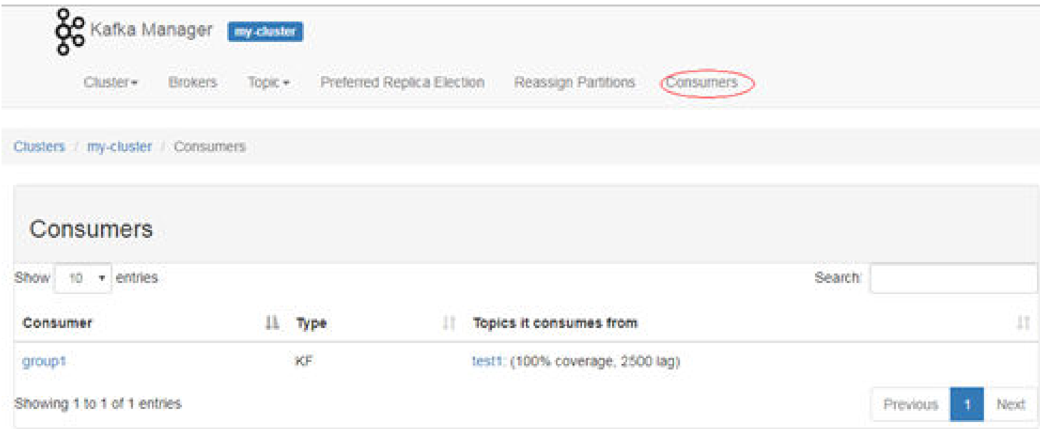
Viewing Consumers Information

Step 1 Log in to the KafkaManager web UI.

Step 2 On the cluster list page, click a cluster name to access the **Summary** page of the cluster.

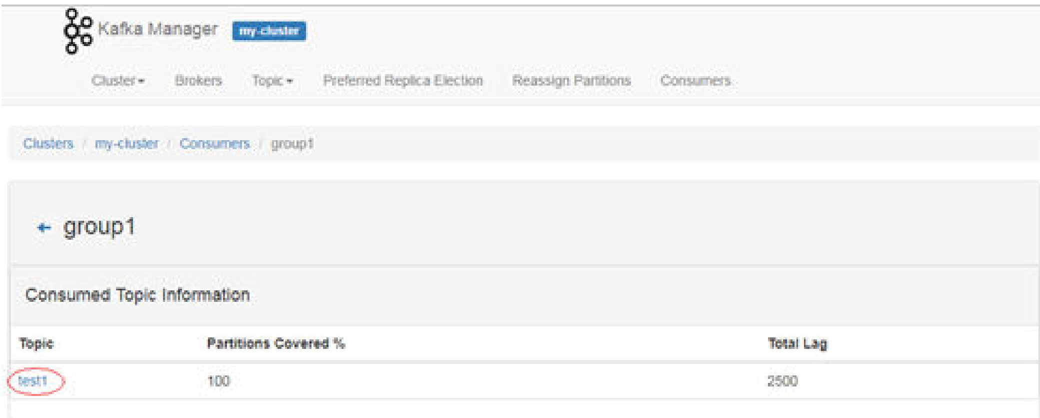
Step 3 Click **Consumers** to view the consumers of the current cluster and each consumer's consumption information.

Figure 16-9 Consumers



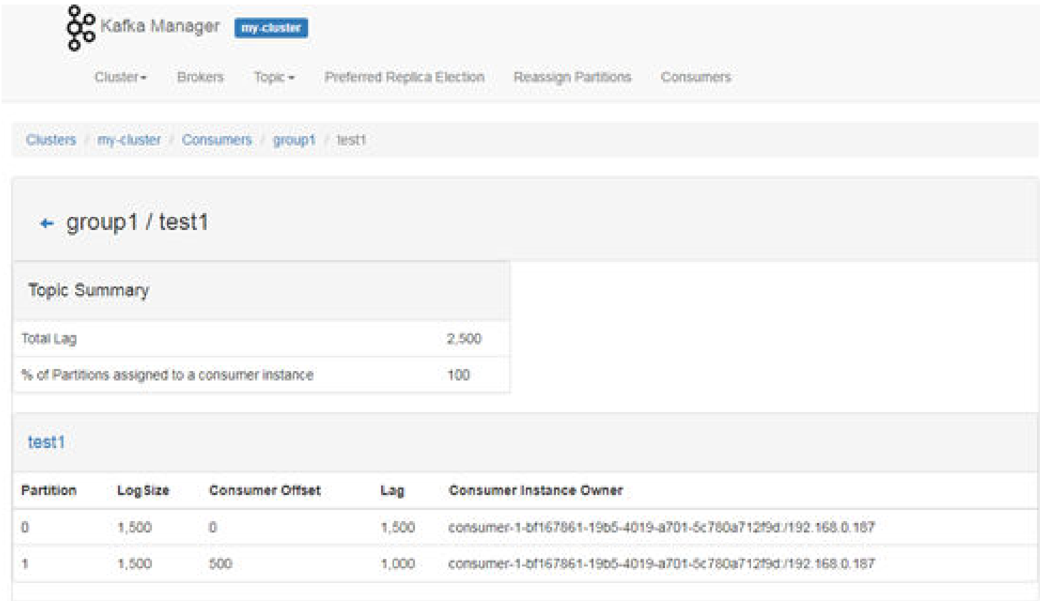
Step 4 Click a consumer name to view the list of the consumed topics.

Figure 16-10 List of topics consumed by the consumer



Step 5 Click a topic name in the topic list of the consumer to view consumption information about the topic.

Figure 16-11 Topic consumption details



----End

Modifying the Partition of a Topic Through KafkaManager

Step 1 Log in to the KafkaManager web UI.

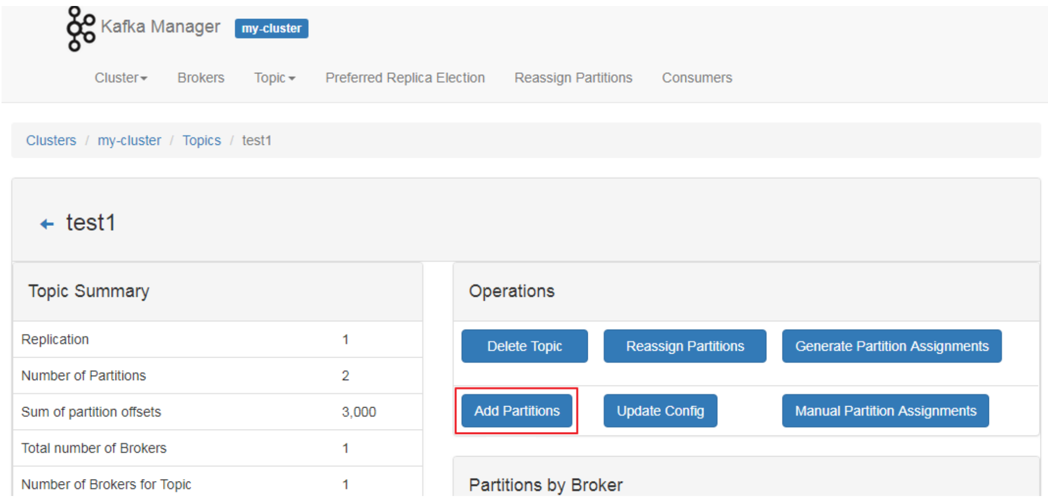
Step 2 On the cluster list page, click a cluster name to access the **Summary** page of the cluster.

Step 3 Choose **Topic** > **List** to access the topic list page of the current cluster.

Step 4 Click a topic name to access the **Topic Summary** page.

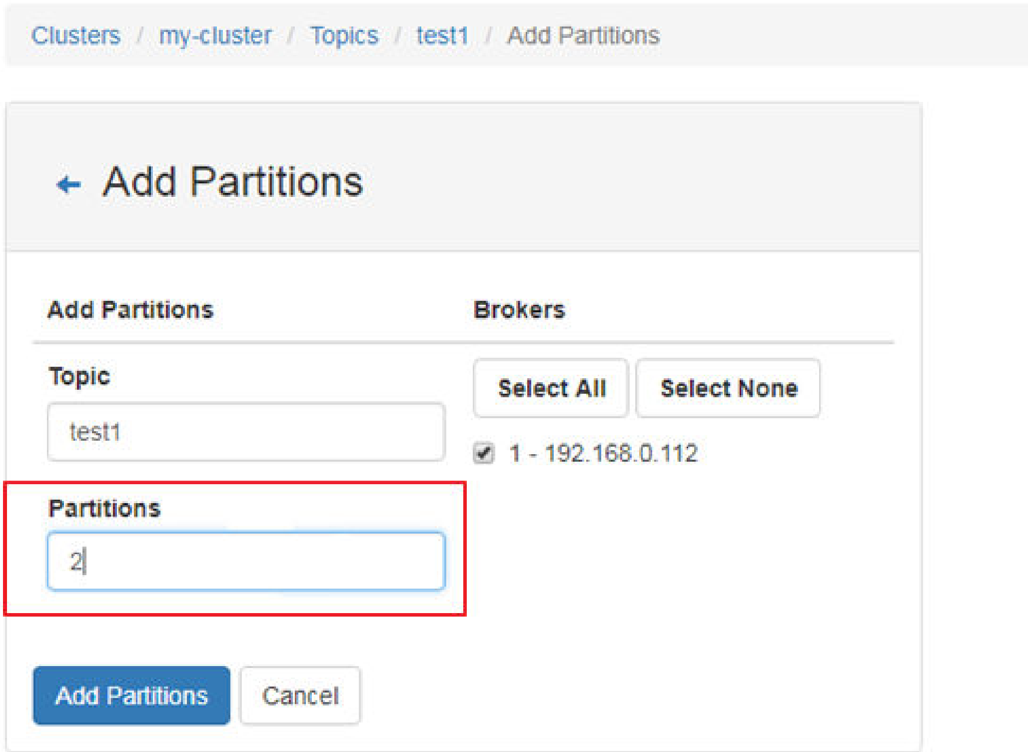
Step 5 Click **Add Partitions**. The page for adding partitions is displayed.

Figure 16-12 Adding partitions



Step 6 Confirm the topic name and modify the value of the **Partitions** parameter and click **Add Partitions** to add partitions.

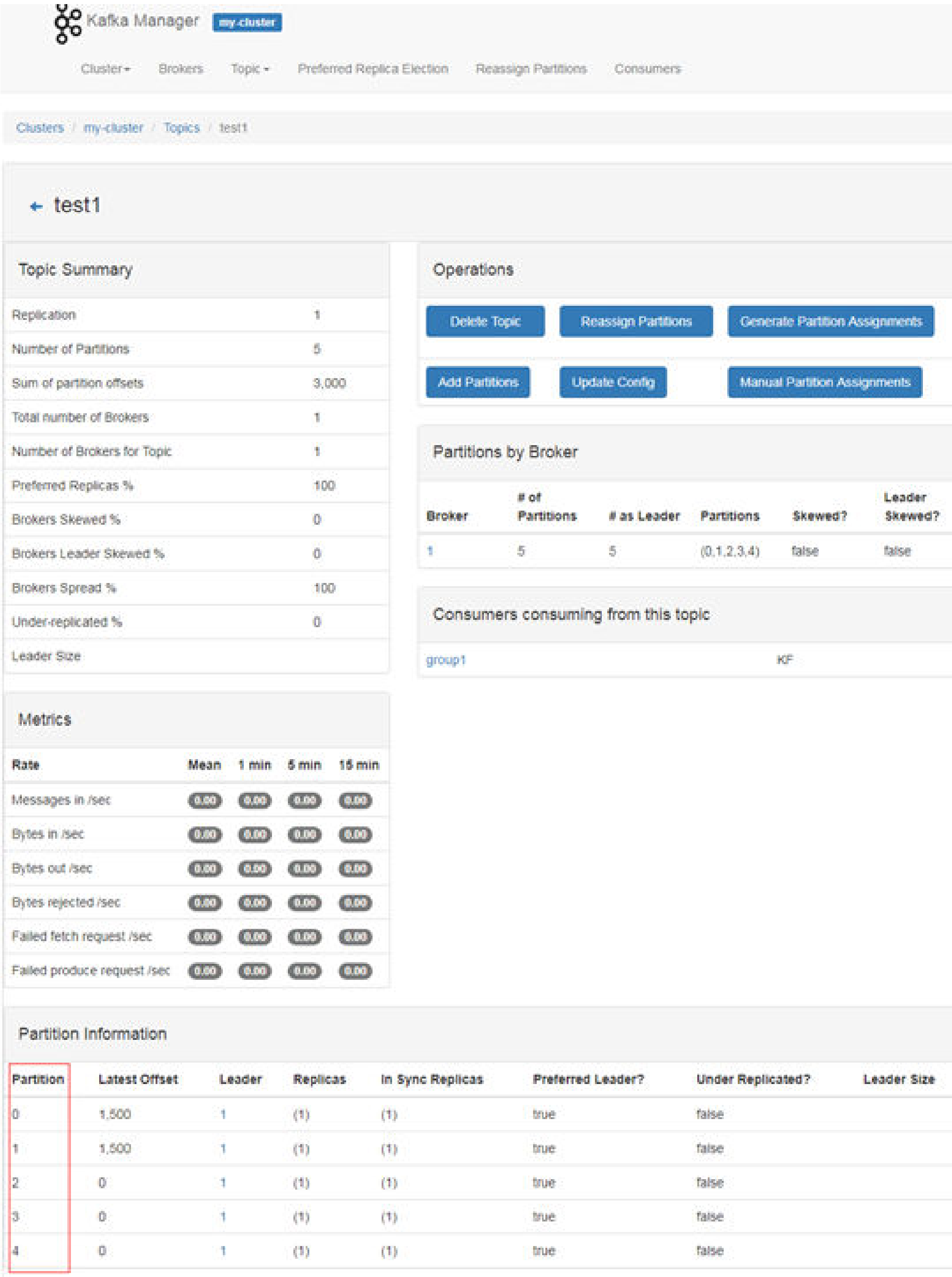
Figure 16-13 Modifying the number of partitions



Step 7 After the partitions are added successfully, click **Go to topic view** to return to the **Topic Summary** page.

Step 8 Check the number of partitions in **Partition Information** in the lower part of the **Topic Summary** page.

Figure 16-14 Partition Information



Step 9 (Optional) If you are not satisfied with the assigned partitions, you can use the partition reassignment function to automatically reassign partitions.

1. On the **Topic Summary** page, click **Generate Partition Assignments**.
2. Select the broker instance and click **Generate Partition Assignments** to generate a partition.
3. After partition generation, click **Go to topic view** to return to the **Topic Summary** page.

4. On the **Topic Summary** page, click **Reassign Partitions** to automatically assign partitions to the broker instance of the cluster.
5. Click **Go to reassign partitions** to view details about the reassigned partitions.

Step 10 (Optional) If you are not satisfied with the automatically assigned partitions, you can manually assign the partitions.

1. On the **Topic Summary** page, click **Manual Partition Assignments** to access the page for manually assign partitions.
2. Manually assign a broker ID to each partition replica, and click **Save Partition Assignment** to save the changes.
3. Click **Go to topic view** to return to the **Topic Summary** page and view the partition details.

----End

17 Using Loader

17.1 Using Loader from Scratch

You can use Loader to import data from the SFTP server to HDFS.

This section applies to MRS clusters earlier than 3.*x*.

Prerequisites

- You have prepared service data.
- You have created an analysis cluster.

Procedure

Step 1 Access the Loader page.

1. Go to the cluster details page and choose **Services**.
2. Choose **Hue**. In **Hue Web UI** of **Hue Summary**, click **Hue (Active)**. The Hue web UI is displayed.
3. Choose **Data Browsers** > **Sqoop**.

   The job management tab page is displayed by default on the Loader page.

Step 2 On the Loader page, click **Manage links**.

Step 3 Click **New link** and create **sftp-connector**. For details, see **File Server Link**.

Step 4 Click **New link**, enter the link name, select **hdfs-connector**, and create **hdfs-connector**.

Step 5 On the Loader page, click **Manage jobs**.

Step 6 Click **New Job**.

Step 7 In **Connection**, set parameters.

1. In **Name**, enter a job name.
2. Select the source link created in **Step 3** and the target link created in **Step 4**.

Step 8 In **From**, configure the job of the source link.

For details, see **ftp-connector or sftp-connector**.

Step 9 In **To**, configure the job of the target link.

For details, see **hdfs-connector**.

Step 10 In **Task Config**, set job running parameters.

Table 17-1 Loader job running properties

Parameter	Description
Extractors	Number of Map tasks
Loaders	Number of Reduce tasks This parameter is displayed only when the destination field is HBase or Hive.
Max. Error Records in a Single Shard	Error record threshold. If the number of error records of a single Map task exceeds the threshold, the task automatically stops and the obtained data is not returned. **NOTE** Data is read and written in batches for **MYSQL** and **MPPDB** of **generic-jdbc-connector** by default. Errors are recorded once at most for each batch of data.
Dirty Data Directory	Directory for saving dirty data. If you leave this parameter blank, dirty data will not be saved.

Step 11 Click **Save**.

----End

17.2 How to Use Loader

This section applies to MRS clusters earlier than 3.*x*.

Process

The process for migrating user data with Loader is as follows:

1. Access the Loader page of the Hue web UI.
2. Manage Loader links.
3. Create a job and select a data source link and a link for saving data.
4. Run the job to complete data migration.

Loader Page

The Loader page is a graphical data migration management tool based on the open source Sqoop web UI and is hosted on the Hue web UI. Perform the following operations to access the Loader page:

1. Access the Hue web UI. For details, see **Accessing the Hue Web UI**.

2. Choose **Data Browsers** > **Sqoop**.

   The job management tab page is displayed by default on the Loader page.

Loader Links

Loader links save data location information. Loader uses links to access data or save data to the specified location. Perform the following operations to access the Loader link management page:

1. Access the Loader page.
2. Click **Manage links**.

   The Loader link management page is displayed.

   Click **Manage jobs** to return to the job management page.
3. Click **New link** to go to the configuration page and set parameters to create a Loader link.

Loader Jobs

Loader jobs are used to manage data migration tasks. Each job consists of a source data link and a destination data link. A job reads data from the source link and saves data to the destination link to complete a data migration task.

17.3 Common Loader Parameters

This section applies to MRS 3.*x* or later.

Navigation Path

For details about the how to set parameters, see **Modifying Cluster Service Configuration Parameters**.

Parameter Description

Table 17-2 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.

17.4 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.

This section applies to MRS 3.*x* or later.

Prerequisites

- The MRS cluster administrator has understood service requirements.
- You have logged in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.

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 17-3**.

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 17-3 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

17.5 Loader Link Configuration

This section applies to versions earlier than MRS 3.x.

Overview

Loader supports the following links. This section describes configurations of each link.

- obs-connector
- generic-jdbc-connector
- ftp-connector or sftp-connector
- hbase-connector, hdfs-connector, or hive-connector

OBS Link

An OBS link is a data exchange channel between Loader and OBS. **Table 17-4** describes the configuration parameters.

Table 17-4 obs-connector configuration

Parameter	Description
Name	Name of a Loader connection
OBS Server	Enter an OBS endpoint. The common format is **OBS.*Region.DomainName***. Run the following command to query the endpoints of OBS: **cat /opt/Bigdata/apache-tomcat-7.0.78/webapps/web/WEB-INF/classes/cloud-obs.properties**
Port	Specifies the port for accessing OBS data. The default value is **443**.
Access Key	AK for a user to access OBS
Security Key	SK corresponding to AK

Relational Database Link

A relational database link is a data exchange channel between Loader and a relational database. **Table 17-5** describes the configuration parameters.

NOTE

Some parameters are hidden by default. They appear only after you click **Show Senior Parameter**.

Table 17-5 generic-jdbc-connector configuration

Parameter	Description
Name	Name of a Loader link
Database Type	Data types supported by Loader links: **ORACLE**, **MYSQL**, and **MPPDB**
Host	Database access address, which can be an IP address or domain name.
Port	Port for accessing the database
Database	Name of the database saving data
Username	Username for accessing the database
Password	Password of the user Use the actual password.

Table 17-6 Senior parameter configuration

Parameter	Description
Fetch Size	A maximum volume of data obtained during each database access
Connection Properties	Drive properties exclusive to the database link supported by databases of different types, for example, **autoReconnect** of MYSQL. If you want to define the drive properties, click **Add**.
Identifier Enclose	Delimiter for reserving keywords in the database SQL. Delimiters defined in different databases vary.

File Server Link

File server links include FTP and SFTP links and serve as a data exchange channel between Loader and a file server. **Table 17-7** describes the configuration parameters.

Table 17-7 ftp-connector or **sftp-connector** configuration

Parameter	Description
Name	Name of a Loader link
Hostname/IP	Enter the file server access address, which can be a host name or IP address.

Parameter	Description
Port	Port for accessing the file server. ● Use port **21** for FTP. ● Use port **22** for SFTP.
Username	Username for logging in to the file server
Password	Password of the user

MRS Cluster Link

MRS cluster links include HBase, HDFS, and Hive links and serve as a data exchange channel between Loader and HBase, HDFS, or Hive.

When configuring an MRS cluster link, set the name, select a connector, for example, **hbase-connector**, **hdfs-connector**, or **hive-connector**, and save the settings.

17.6 Managing Loader Links (Versions Earlier Than MRS 3.x)

Scenario

You can create, view, edit, and delete links on the Loader page.

This section applies to versions earlier than MRS 3.x.

Prerequisites

You have accessed the Loader page. For details, see **Loader Page**.

Creating a Link

Step 1 On the Loader page, click **Manage links**.

Step 2 Click **New link** and configure link parameters.

For details about the parameters, see **Loader Link Configuration**.

Step 3 Click **Save**.

If link configurations, for example, IP address, port, and access user information, are incorrect, the link will fail to be verified and saved. In addition, VPC configurations may affect the network connectivity.

NOTE

You can click **Test** to immediately check whether the link is available.

----End

Viewing a Link

Step 1 On the Loader page, click **Manage links**.

- If Kerberos authentication is enabled for the cluster, all links created by the current user are displayed by default and other users' links cannot be displayed.
- If Kerberos authentication is disabled for the cluster, all Loader links of the cluster are displayed.

Step 2 In **Sqoop Links**, enter a link name to filter the link.

----End

Editing a Link

Step 1 On the Loader page, click **Manage links**.

Step 2 Click the link name to go to the edit page.

Step 3 Modify the link configuration parameters based on service requirements.

Step 4 Click **Test**.

If the test is successful, go to **Step 5**. If a message displays indicating that OBS server cannot be connected, repeat **Step 3**.

Step 5 Click **Save**.

If a Loader job has integrated into a Loader link, editing the link parameters may affect Loader running.

----End

Deleting a Link

Step 1 On the Loader page, click **Manage links**.

Step 2 Locate the row that contains the target link, and click **Delete**.

Step 3 In the dialog box, click **Yes, delete it**.

If a Loader job has integrated a Loader link, the link cannot be deleted.

----End

17.7 Managing Loader Links (MRS 3.*x* and Later Versions)

Scenario

You can create, view, edit, and delete links on the Loader page.

This section applies to MRS 3.*x* or later.

Creating a Connection

Step 1 Log in to the service page.

For MRS 3.*x* or later, log in to FusionInsight Manager (for details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**) 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 17-8**.
- **ftp-connector**: For details about parameter settings, see **Table 17-9**.
- **sftp-connector**: For details about parameter settings, see **Table 17-10**.
- **hdfs-connector**: For details about parameter settings, see **Table 17-11**.
- **oracle-connector**: For details about parameter settings, see **Table 17-12**.
- **mysql-fastpath-connector**: For details about parameter settings, see **Table 17-14**.
- **oracle-partition-connector**: For details about parameter settings, see **Table 17-13**.

Table 17-8 generic-jdbc-connector configuration

Parameter	Description
Name	Name of a Loader connection
Connector	Select **generic-jdbc-connector**.
JDBC Driver Class	JDBC driver classes are as follows: • oracle: **oracle.jdbc.driver.OracleDriver** • SQLServer: **com.microsoft.sqlserver.jdbc.SQLServerDriver** • mysql: **com.mysql.jdbc.Driver** • postgresql: **org.postgresql.Driver** • gaussdb200: **com.huawei.gauss200.jdbc.Driver**

Parameter	Description
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: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 17-9 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 17-10 sftp-connector configuration

Parameter	Description
Name	Name of a Loader connection
Connector	Select **sftp-connector**.

Table 17-11 hdfs-connector configuration

Parameter	Description
Name	Name of a Loader connection
Connector	Select **hdfs-connector**.

Table 17-12 oracle-connector configuration

Parameter	Description
Name	Name of a Loader connection
Connector	Select **oracle-connector**.
JDBC Connection String	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 17-13 oracle-partition-connector configuration

Parameter	Description
Name	Name of a Loader connection
Connector	Select **oracle-partition-connector**.
JDBC Driver Class	Enter **oracle.jdbc.driver.OracleDriver**.
JDBC Connection String	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 17-14 mysql-fastpath-connector configuration

Parameter	Description
Name	Name of a Loader connection

Parameter	Description
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	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.

17.8 Source Link Configurations of Loader Jobs

Overview

When Loader jobs obtain data from different data sources, a link corresponding to a data source type needs to be selected and the link properties need to be configured.

This section applies to versions earlier than MRS 3.x.

obs-connector

Table 17-15 Data source link properties of **obs-connector**

Parameter	Description
Bucket Name	OBS file system for storing source data.
Source Directory/ File	Actual storage form of source data. It can be either all data files in a directory or a single data file contained in the file system.

Parameter	Description
File Format	Loader supports the following file formats of data stored in OBS: ● **CSV_FILE**: Specifies a text file. When the destination link is a database link, only the text file is supported. ● **BINARY_FILE**: Specifies binary files excluding text files.
Line Separator	Identifier of each line end of source data
Field Separator	Identifier of each field end of source data
Encoding Type	Text encoding type of source data. It takes effect on text files only.
File Split Type	The following types are supported: ● **File**: The number of files is assigned to a map task by the total number of files. The calculation formula is **Total number of files/Extractors**. ● **Size**: A file size is assigned to a map task by the total file size. The calculation formula is **Total file size/ Extractors**.

generic-jdbc-connector

Table 17-16 Data source link properties of **generic-jdbc-connector**

Parameter	Description
Schema/ Tablespace	Name of the database storing source data. You can query and select it on the interface.
Table Name	Data table storing the source data. You can query and select it on the interface.
Partition Column	If multiple columns need to be read, use this column to split the result and obtain data.
Where Clause	Query statement used when accessing the database

ftp-connector or sftp-connector

Table 17-17 Data source link properties of **ftp-connector** or **sftp-connector**

Parameter	Description
Source Directory/ File	Actual storage form of source data. It can be either all data files in a directory or single data file contained in the file server.

Parameter	Description
File Format	Loader supports the following file formats of data stored in the file server: ● **CSV_FILE**: Specifies a text file. When the destination link is a database link, only the text file is supported. ● **BINARY_FILE**: Specifies binary files excluding text files.
Line Separator	Identifier of each line end of source data **NOTE** If FTP or SFTP serves as a source link and **File Format** is set to **BINARY_FILE**, the value of **Line Separator** in the advanced properties is invalid.
Field Separator	Identifier of each field end of source data **NOTE** If FTP or SFTP serves as a source link and **File Format** is set to **BINARY_FILE**, the value of **Field Separator** in the advanced properties is invalid.
Encoding Type	Text encoding type of source data. It takes effect on text files only.
File Split Type	The following types are supported: ● **File**: The number of files is assigned to a map task by the total number of files. The calculation formula is **Total number of files/Extractors**. ● **Size**: A file size is assigned to a map task by the total file size. The calculation formula is **Total file size/ Extractors**.

hbase-connector

Table 17-18 Data source link properties of **hbase-connector**

Parameter	Description
Table Name	HBase table storing source data

hdfs-connector

Table 17-19 Data source link properties of **hdfs-connector**

Parameter	Description
Source Directory/ File	Actual storage form of source data. It can be either all data files in a directory or single data file contained in HDFS.

Parameter	Description
File Format	Loader supports the following file formats of data stored in HDFS: ● **CSV_FILE**: Specifies a text file. When the destination link is a database link, only the text file is supported. ● **BINARY_FILE**: Specifies binary files excluding text files.
Line Separator	Identifier of each line end of source data **NOTE** If HDFS serves as a source link and **File Format** is set to **BINARY_FILE**, the value of **Line Separator** in the advanced properties is invalid.
Field Separator	Identifier of each field end of source data **NOTE** If HDFS serves as a source link and **File Format** is set to **BINARY_FILE**, the value of **Field Separator** in the advanced properties is invalid.
File Split Type	The following types are supported: ● **File**: The number of files is assigned to a map task by the total number of files. The calculation formula is **Total number of files/Extractors**. ● **Size**: A file size is assigned to a map task by the total file size. The calculation formula is **Total file size/ Extractors**.

hive-connector

Table 17-20 Data source link properties of **hive-connector**

Parameter	Description
Database Name	Name of the Hive database storing the data source. You can query and select it on the interface.
Table	Name of the Hive table storing the data source. You can query and select it on the interface.

17.9 Destination Link Configurations of Loader Jobs

Overview

When Loader jobs save data to different storage locations, a destination link needs to be selected and the link properties need to be configured.

obs-connector

Table 17-21 Destination link properties of **obs-connector**

Parameter	Description
Bucket Name	OBS file system for storing final data.
Output Directory	Directory for storing final data in the file system. A directory must be specified.
File Format	Loader supports the following file formats of data stored in OBS: ● **CSV_FILE**: Specifies a text file. When the destination link is a database link, only the text file is supported. ● **BINARY_FILE**: Specifies binary files excluding text files.
Line Separator	Identifier of each line end of final data
Field Separator	Identifier of each field end of final data
Encoding Type	Text encoding type of final data. It takes effect on text files only.

generic-jdbc-connector

Table 17-22 Destination link properties of **generic-jdbc-connector**

Parameter	Description
Schema Name	Name of the database storing final data
Table	Name of the table saving final data

ftp-connector or sftp-connector

Table 17-23 Destination link properties of **ftp-connector** or **sftp-connector**

Parameter	Description
Output Directory	Directory for storing final data in the file server. A directory must be specified.
File Format	Loader supports the following file formats of data stored in the file server: ● **CSV_FILE**: Specifies a text file. When the destination link is a database link, only the text file is supported. ● **BINARY_FILE**: Specifies binary files excluding text files.

Parameter	Description
Line Separator	Identifier of each line end of final data **NOTE** If FTP or SFTP serves as a destination link and **File Format** is set to **BINARY_FILE**, the value of **Line Separator** in the advanced properties is invalid.
Field Separator	Identifier of each field end of final data **NOTE** If FTP or SFTP serves as a destination link and **File Format** is set to **BINARY_FILE**, the value of **Field Separator** in the advanced properties is invalid.
Encoding Type	Text encoding type of final data. It takes effect on text files only.

hbase-connector

Table 17-24 Destination link properties of **hbase-connector**

Parameter	Description
Table Name	Name of the HBase table saving final data. You can query and select it on the interface.
Method	Data can be imported to an HBase table using either **BULKLOAD** or **PUTLIST**.
Clear Data Before Import	Whether to clear data in the destination HBase table. Options are as follows: ● **True**: Clean up data in the table. ● **False**: Do not clean up data in the table. If you select **False**, an error is reported during job running if data exists in the table.

hdfs-connector

Table 17-25 Destination link properties of **hdfs-connector**

Parameter	Description
Output Directory	Directory for storing final data in HDFS. A directory must be specified.

Parameter	Description
File Format	Loader supports the following file formats of data stored in HDFS: ● **CSV_FILE**: Specifies a text file. When the destination link is a database link, only the text file is supported. ● **BINARY_FILE**: Specifies binary files excluding text files.
Compression Codec	Compression mode used when a file is saved to HDFS. The following modes are supported: **NONE**, **DEFLATE**, **GZIP**, **BZIP2**, **LZ4**, and **SNAPPY**.
Overwrite	How to process files in the output directory when files are imported to HDFS. Options are as follows: ● **True**: Clean up files in the directory and import new files by default. ● **False**: Do not clean up files. If files exist in the output directory, job running fails.
Line Separator	Identifier of each line end of final data **NOTE** If HDFS serves as a destination link and **File Format** is set to **BINARY_FILE**, the value of **Line Separator** in the advanced properties is invalid.
Field Separator	Identifier of each field end of final data **NOTE** If HDFS serves as a destination link and **File Format** is set to **BINARY_FILE**, the value of **Field Separator** in the advanced properties is invalid.

hive-connector

Table 17-26 Destination link properties of **hive-connector**

Parameter	Description
Database	Name of the Hive database storing final data. You can query and select it on the interface.
Table	Name of the Hive table saving final data. You can query and select it on the interface.

17.10 Managing Loader Jobs

Scenario

You can create, view, edit, and delete jobs on the Loader page.

This section applies to versions earlier than MRS 3.x.

Prerequisites

You have accessed the Loader page. For details, see **Loader Page**.

Creating a Job

Step 1 On the Loader page, click **New job**.

Step 2 In **Connection**, set parameters.

1. In **Name**, enter a job name.
2. In **From link** and **To link**, select links accordingly.

   After you select a link of a type, data is obtained from the specified source and saved to the destination.

   NOTE

   If no available link exists, click **Add a new link**.

Step 3 In **From**, configure the job of the source link.

For details, see **Source Link Configurations of Loader Jobs**.

Step 4 In **To**, configure the job of the destination link.

For details, see **Destination Link Configurations of Loader Jobs**.

Step 5 Check whether a database link is selected in **To link**.

Database links include:

- generic-jdbc-connector
- hbase-connector
- hive-connector

If you set **To link** to a database link, you need to configure a mapping between service data and a field in the database table.

- If you set it to a database link, go to **Step 6**.
- If you do not set it to a database link, go to **Step 7**.

Step 6 In **Field Mapping**, enter a field mapping. Then proceed to **Step 7**.

Field Mapping specifies a mapping between each column of user data and a field in the database table.

Table 17-27 Field Mapping properties

Parameter	Description
Column Num	Field sequence of service data
Sample	First row of sample values of service data
Column Family	When **To link** is **hbase-connector**, you can select a column family for storing data.
Destination Field	Field for storing data

Parameter	Description
Type	Type of the field selected by the user
Row Key	When **To link** is **hbase-connector**, you need to select **Destination Field** as a row key.

NOTE

If the value of **From** is a connector of a file type, for example, SFTP, FTP, OBS, and HDFS files, the value of **Field Mapping** is the first row of data in the file. Ensure that the first row of data is complete. Otherwise, the Loader job will not extract columns that are not mapped.

Step 7 In **Task Config**, set job running parameters.

Table 17-28 Loader job running properties

Parameter	Description
Extractors	Number of Map tasks
Loaders	Number of Reduce tasks This parameter is displayed only when the destination field is HBase or Hive.
Max. Error Records in a Single Shard	Error record threshold. If the number of error records of a single Map task exceeds the threshold, the task automatically stops and the obtained data is not returned. **NOTE** Data is read and written in batches for **MYSQL** and **MPPDB** of **generic-jdbc-connector** by default. Errors are recorded once at most for each batch of data.
Dirty Data Directory	Specifies the directory for saving dirty data. If you leave this parameter blank, dirty data will not be saved.

Step 8 Click **Save**.

----End

Viewing a Job

Step 1 Access the Loader page. The Loader job management page is displayed by default.

- If Kerberos authentication is enabled for the cluster, all jobs created by the current user are displayed by default and other users' jobs cannot be displayed.
- If Kerberos authentication is disabled for the cluster, all Loader jobs of the cluster are displayed.

Step 2 In **Sqoop Jobs**, enter a job name to filter the job.

Step 3 Click **Refresh** to obtain the latest job status.

----End

Editing a Job

Step 1 Access the Loader page. The Loader job management page is displayed by default.

Step 2 Click the job name to go to the edit page.

Step 3 Modify the job configuration parameters based on service requirements.

Step 4 Click **Save**.

NOTE

Basic job operations in the navigation bar on the left are **Run**, **Copy**, **Delete**, **Disable**, **History Record**, and **Show Job JSON Definition**.

----End

Deleting a Job

Step 1 Access the Loader page.

Step 2 In the row of the specified job, click ✖.

You can also select one or more jobs and click **Delete** Job in the upper right corner of the job list.

Step 3 In the dialog box, click **Yes, delete it**.

If the state of a Loader job is **Running**, the job fails to be deleted.

----End

17.11 Preparing a Driver for MySQL Database Link

Scenario

As a component for batch data export, Loader can import and export data using a relational database.

Prerequisites

You have prepared service data.

Procedure

Procedure for MRS clusters earlier than 3.*x*:

Step 1 Download the MySQL JDBC driver **mysql-connector-java-5.1.21.jar** from the MySQL official website. For details about how to select the MySQL JDBC driver, see the following table.

Table 17-29 Version information

JDBC Driver Version	MySQL Version
Connector/J 5.1	MySQL 4.1, MySQL 5.0, MySQL 5.1, and MySQL 6.0 alpha
Connector/J 5.0	MySQL 4.1, MySQL 5.0 servers, and distributed transaction (XA)
Connector/J 3.1	MySQL 4.1, MySQL 5.0 servers, and MySQL 5.0 except distributed transaction (XA)
Connector/J 3.0	MySQL 3.x and MySQL 4.1

Step 2 Upload **mysql-connector-java-5.1.21.jar** to the Loader installation directory on the active and standby MRS Master nodes.

- For versions earlier than MRS 3.x, upload the package to **/opt/Bigdata/MRS_XXX/install/FusionInsight-Sqoop-1.99.7/FusionInsight-Sqoop-1.99.7/server/jdbc/**.

  In the preceding path, **XXX** indicates the MRS version number. Change it based on site requirements.

Step 3 Change the owner of the **mysql-connector-java-5.1.21.jar** package to **omm:wheel**.

Step 4 Modify the **jdbc.properties** configuration file.

Change the key value of **MYSQL** to **mysql-connector-java-5.1.21.jar**, for example, **MYSQL=mysql-connector-java-5.1.21.jar**.

Step 5 Restart the Loader service.

----End

Procedure for MRS cluster 3.*x* and later versions:

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_8.1.0.1/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib**

NOTE

The version 8.1.0.1 is used as an example. Replace it with the actual version number.

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_8.1.0.1/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

17.12 Importing Data

17.12.1 Overview

The section "Importing Data" applies to MRS 3.*x* or later.

Description

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

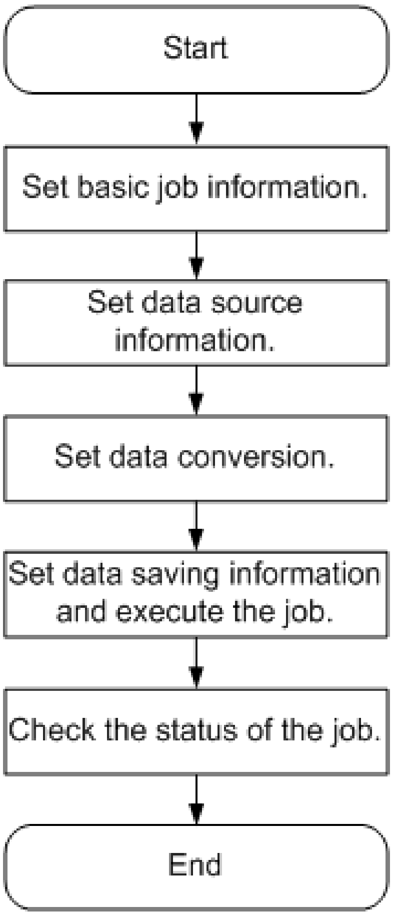
- When an FTP data source connector is used, data is not encrypted. This may result in security risks. You are advised to use an SFTP data source connector.
- You are advised to deploy the SFTP server, FTP server, database server, and Loader into separate subnets to ensure secure data import.
- For connection 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.

Import Process

You can import data on the Loader web UI. **Figure 17-1** shows the data import process.

Figure 17-1 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.

17.12.2 Importing Data Using Loader

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_8.1.0.1/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib**.

   NOTE

   The version 8.1.0.1 is used as an example. Replace it with the actual version number.

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_8.1.0.1/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

   chown omm:wheel *JAR package name*

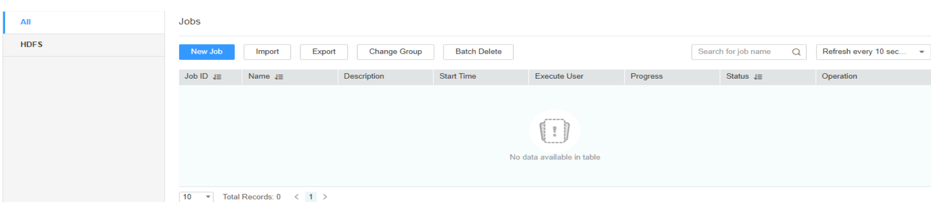
   chmod 600 *JAR package name*

3. Log in to FusionInsight Manager. Choose **Cluster** > *Name of the desired 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 (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-2 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 17-30 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
	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.

Connector	Parameter	Description
	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
sftp-connector	SFTP Server IP Address	IP address of the SFTP server

Connector	Parameter	Description
	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 17-31 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 17-32 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 **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 17-33 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.

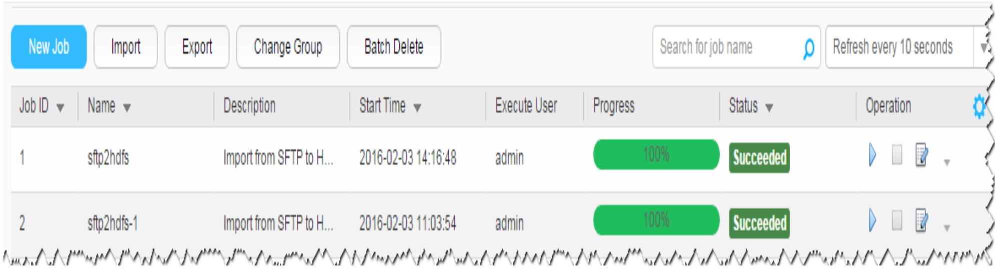
Storage Type	Parameter	Description
	Compression Format	Compression format of files imported to HDFS. Select a format from the drop-down list. If you select **NONE** or do not set this parameter, data is not compressed.
	Output Directory	Directory for storing data imported into HDFS.
	Operation	Action 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 17-3 Viewing Loader jobs



----End

17.12.3 Typical Scenario: Importing 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.

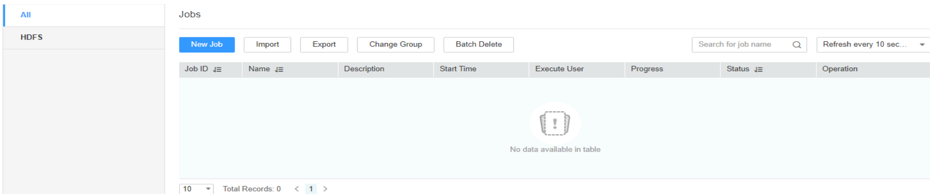
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-4 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-5 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 17-34 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 17-35 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

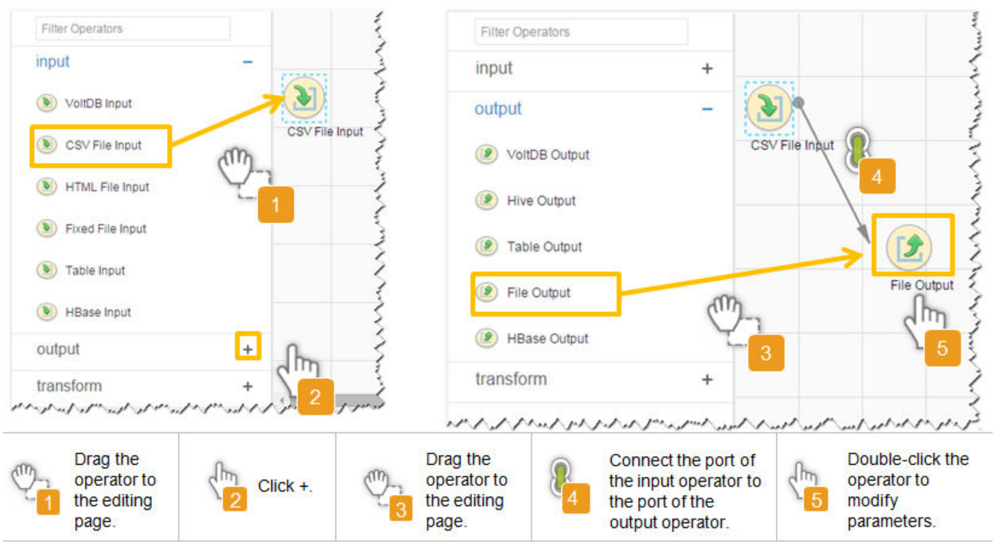
Setting Data Transformation

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 **Operator Help** and **Table 17-36**.

Table 17-36 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 17-6 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 17-37 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

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. 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

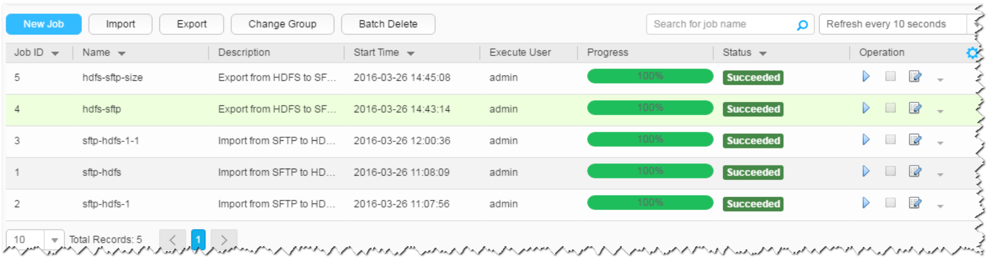
Parameter	Description	Example Value
Extractor Size	Size of data processed by Maps that are started in a MapReduce task of a data configuration operation. The unit is MB. The value must be greater than or equal to 100. The recommended value is 1000. This parameter cannot be set when **Extractors** is 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 17-7 Viewing job details



----End

17.12.4 Typical Scenario: Importing 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.

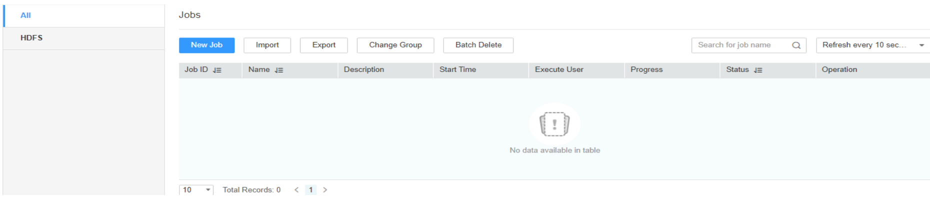
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-8 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-9 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 17-38 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 17-39 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

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 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
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*

Parameter	Description	Example Value
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
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

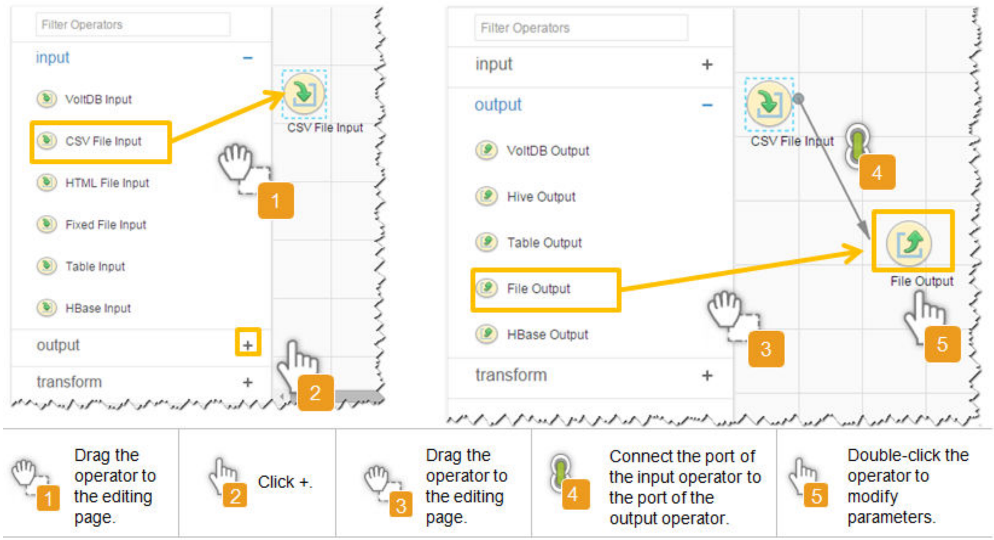
Setting Data Transformation

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 **Operator Help** and **Table 17-40**.

Table 17-40 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 17-10 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 17-41 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. 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

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 SFTP 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

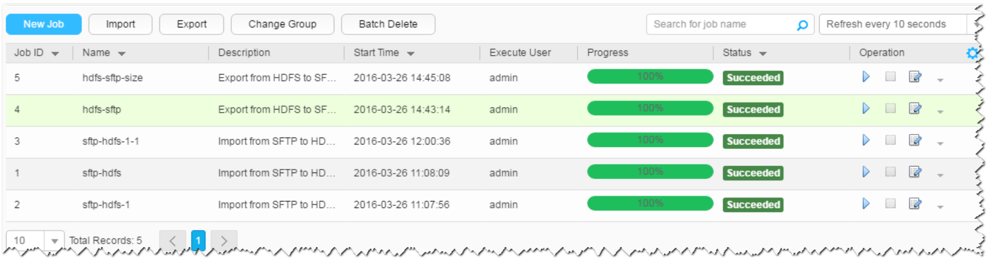
Storage Type	Applicable Scenario	Parameter	Description	Example Value
		Extractors	Number of Maps that are started at the same time in a MapReduce task of a data configuration operation. The value must be less than or equal to 3000.	20
		Extractor Size	HBase does not support this parameter. Please set **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 17-11 Viewing job details



----End

17.12.5 Typical Scenario: Importing 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.

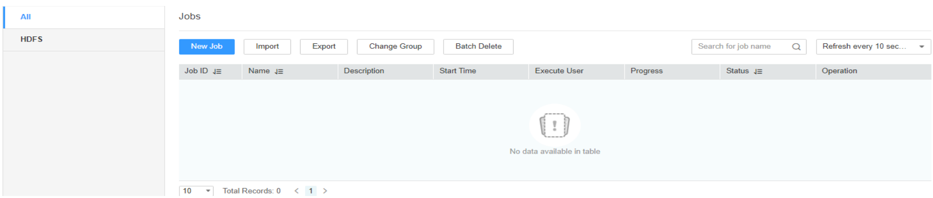
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-12 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-13 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 17-42 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 Hive.

Setting Data Source Information

Step 4 Click **Next**. On the displayed **From** page, set the data source information.

Table 17-43 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
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

Parameter	Description	Example Value
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

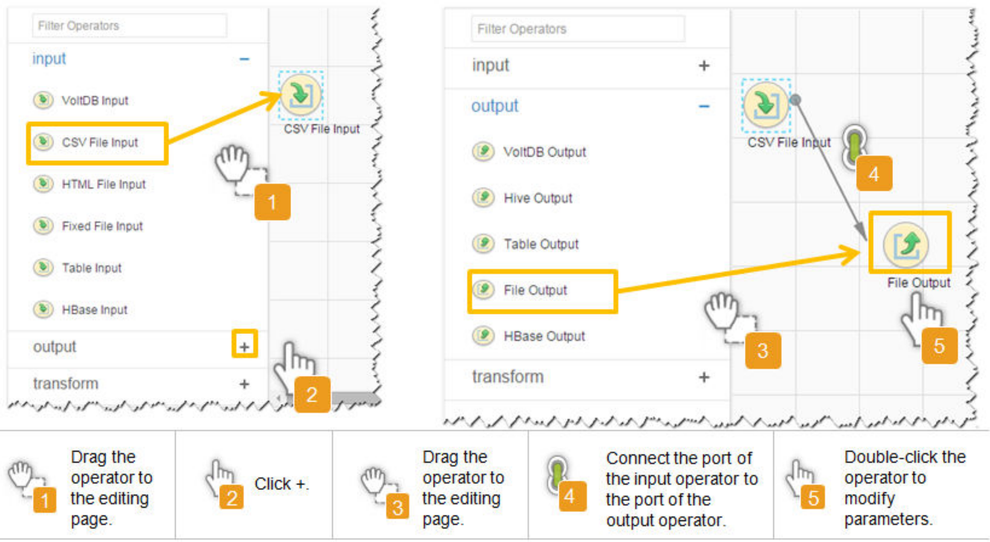
Setting Data Transformation

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 **Operator Help** and **Table 17-44**.

Table 17-44 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 17-14 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 17-45 Parameter description

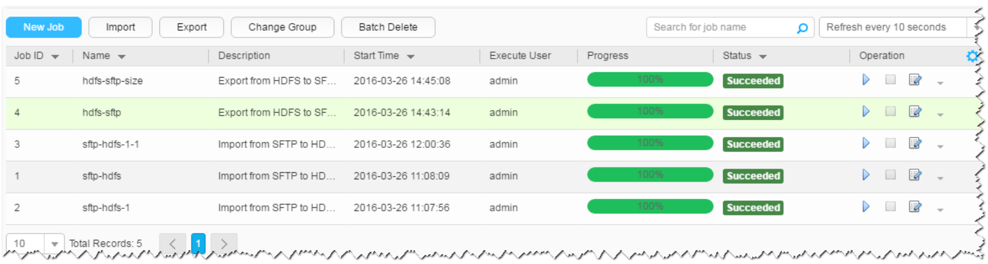
Parameter	Description	Example Value
Output Directory	Directory for storing data imported into Hive. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/opt/tempfile
Extractors	Number of Maps that are started at the same time in 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 17-15 Viewing job details



----End

17.12.6 Typical Scenario: Importing 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.

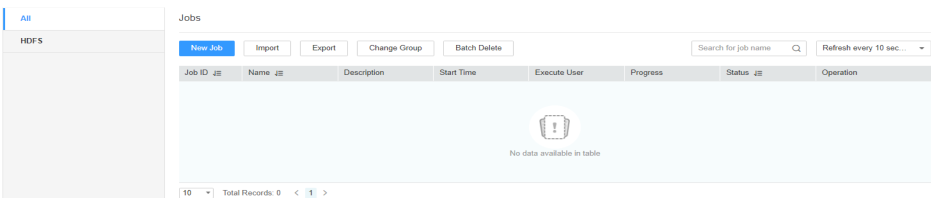
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name*, **Active)**. The Loader web UI is displayed.

Figure 17-16 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-17 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 17-46 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
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

Parameter	Description	Example Value
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 17-47 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

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 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
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*

Parameter	Description	Example Value
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
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

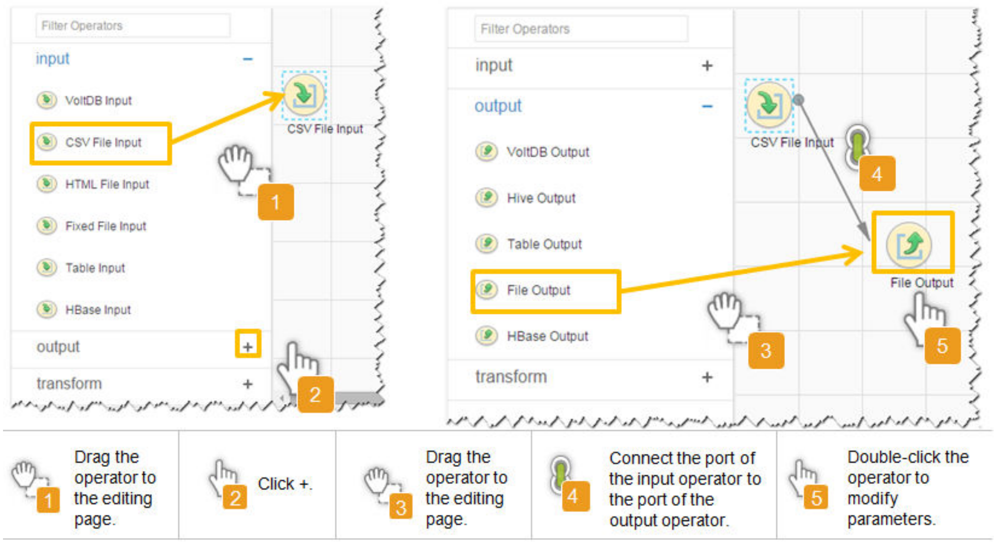
Setting Data Transformation

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 **Operator Help** and **Table 17-48**.

Table 17-48 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 17-18 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 17-49 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

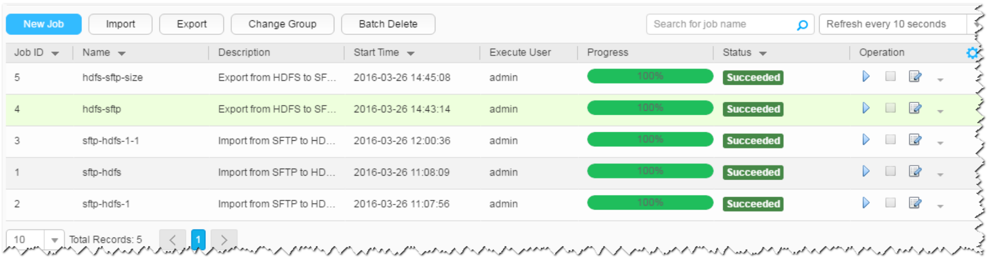
Storage Type	Applicable Scenario	Parameter	Description	Example Value
		Clear data before import	Indicates whether to clear data in the original table before importing data. **True** indicates clearing data and **False** indicates not to clear data. If you do not set this parameter, the original table 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. 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 17-19 Viewing job details



----End

17.12.7 Typical Scenario: Importing 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_8.1.0.1/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_8.1.0.1/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** > *Name of the desired cluster* > **Services** > **Loader** > **More** > **Restart Service**. Enter the password of the administrator to restart the Loader service.

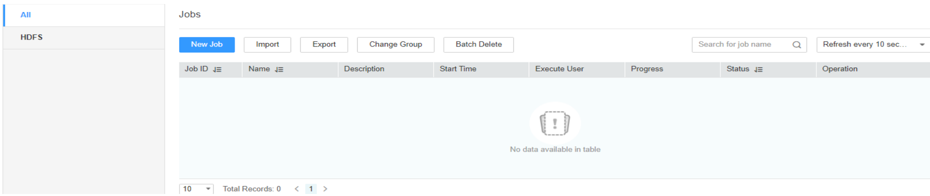
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-20 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-21 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 17-50 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 17-51 Parameter description

Parameter	Description	Example Value
Schema Name	Database schema name. This parameter exists in the **Table name** schema.	public

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 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

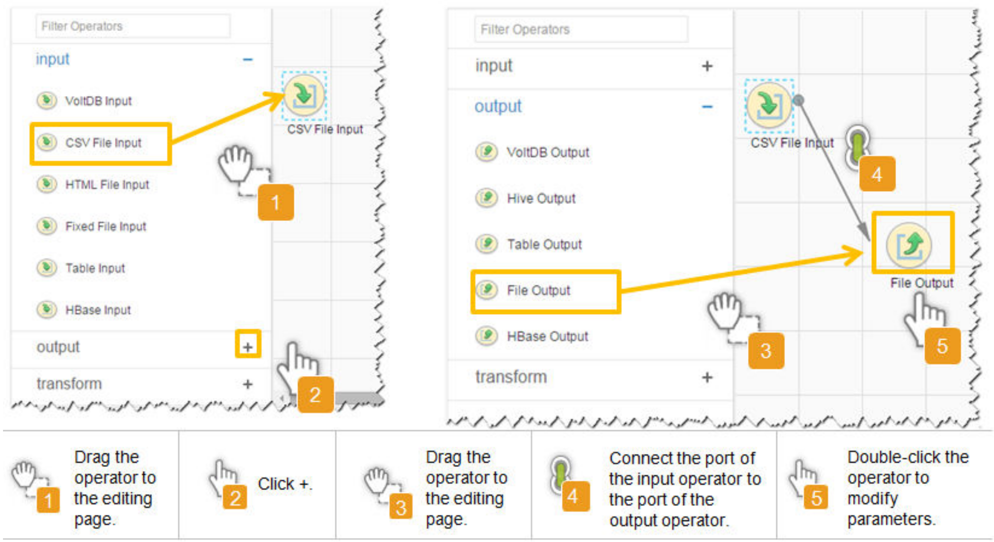
Setting Data Transformation

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 **Operator Help** and **Table 17-52**.

Table 17-52 Input and output parameters of the operator

Input Type	Output Type
Table input	File output

Figure 17-22 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 17-53 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

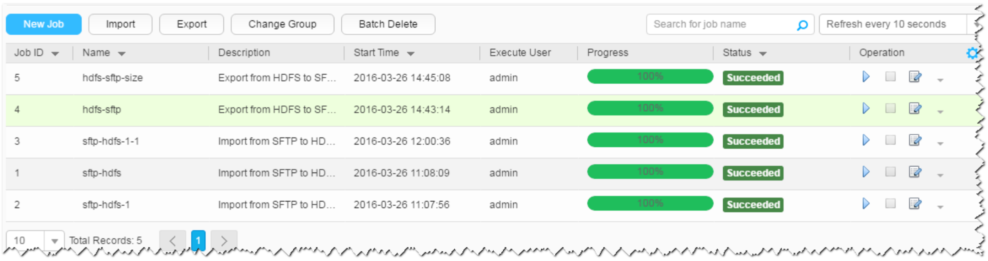
Parameter	Description	Example Value
Operation	Action during data import. When all data is to be imported from the input path to the destination path, the data is stored in a temporary directory and then copied from the temporary directory to the destination path. After the data is imported successfully, 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 17-23 Viewing job details



----End

17.12.8 Typical Scenario: Importing 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_8.1.0.1/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_8.1.0.1/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** > *Name of the desired cluster* > **Services** > **Loader** > **More** > **Restart Service**. Enter the password of the administrator to restart the Loader service.

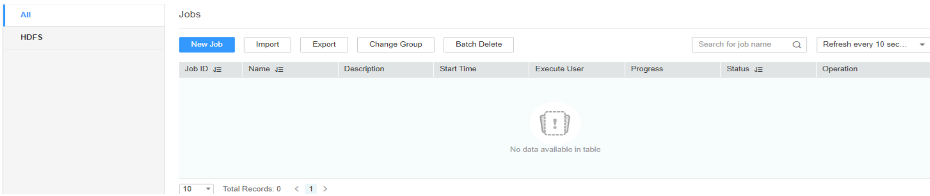
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-24 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-25 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 17-54 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 17-55 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.	-

<table>
<tr><th>Parameter</th><th>Description</th><th>Example Value</th></tr>
<tr><td>Partition Column Name</td><td>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.</td><td>id</td></tr>
<tr><td>Nulls in Partition Column</td><td>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.</td><td>true</td></tr>
<tr><td>Whether to Specify a Partition Column</td><td>Indicates whether to specify a partition column.</td><td>true</td></tr>
</table>

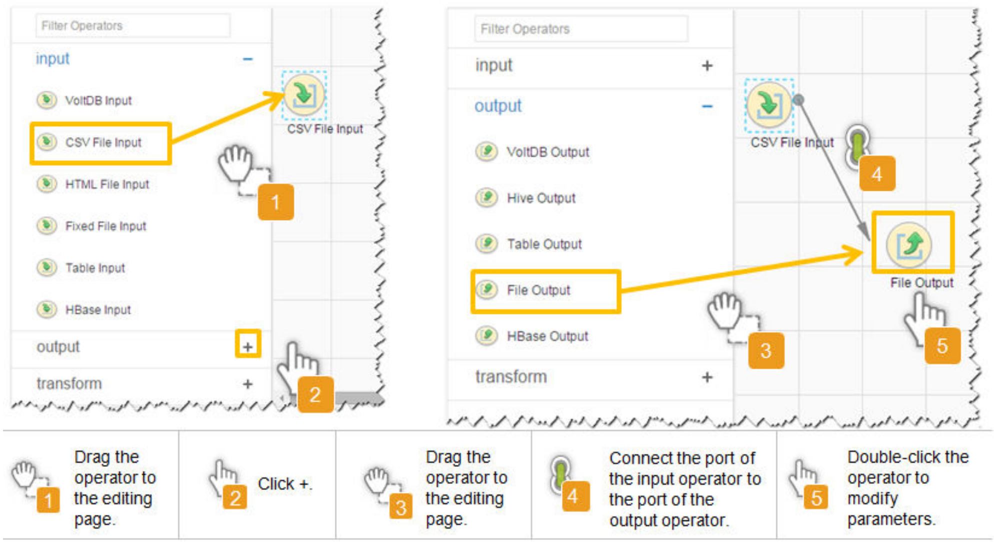
Setting Data Transformation

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 **Operator Help** and **Table 17-56**.

Table 17-56 Input and output parameters of the operator

Input Type	Output Type
Table input	HBase output

Figure 17-26 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 17-57 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

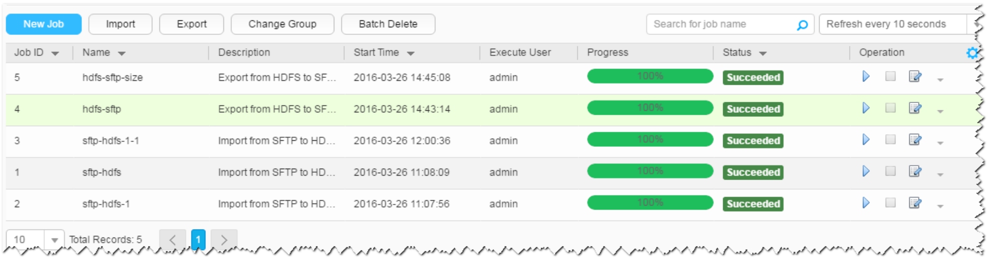
Storage Type	Applicable Scenario	Parameter	Description	Example Value
		Clear data before import	Indicates whether to clear data in the original table before importing data. **True** indicates clearing data and **False** indicates not to clear data. If you do not set this parameter, the original table 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.	true
		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 17-27 Viewing job details



----End

17.12.9 Typical Scenario: Importing 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_8.1.0.1/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_8.1.0.1/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** > *Name of the desired cluster* > **Services** > **Loader** > **More** > **Restart Service**. Enter the password of the administrator to restart the Loader service.

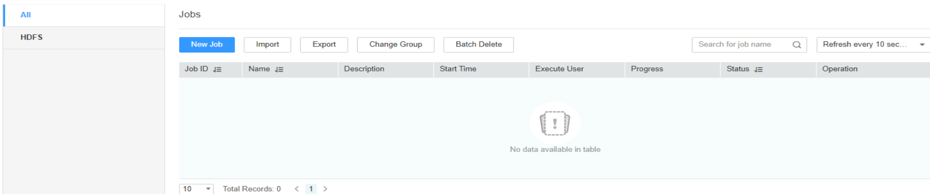
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-28 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-29 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 17-58 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 17-59 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

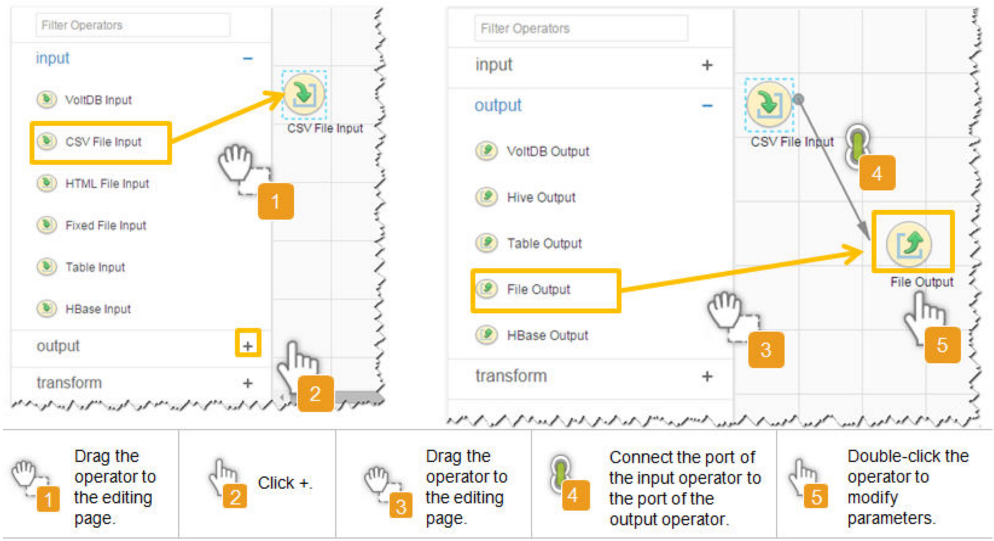
Setting Data Transformation

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 **Operator Help** and **Table 17-60**.

Table 17-60 Input and output parameters of the operator

Input Type	Output Type
Table input	Hive output

Figure 17-30 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 17-61 Parameter description

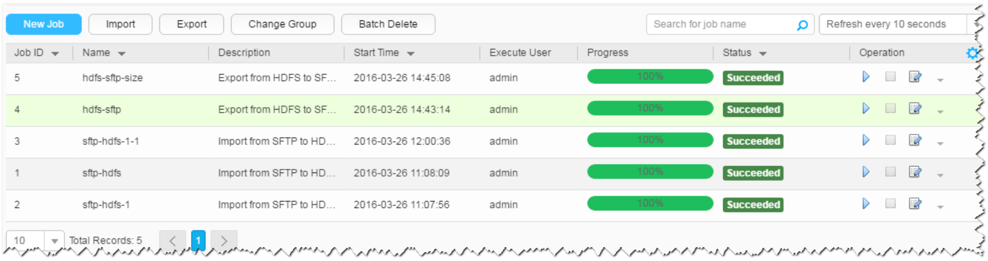
Parameter	Description	Example Value
Output Directory	Directory for storing data imported into Hive. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/opt/tempfile
Extractors	Number of Maps that are started at the same time in 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 17-31 Viewing job details



----End

17.12.10 Typical Scenario: Importing 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

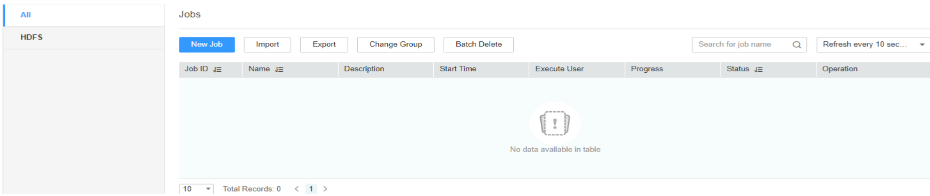
Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.

2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-32 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-33 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 17-62 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

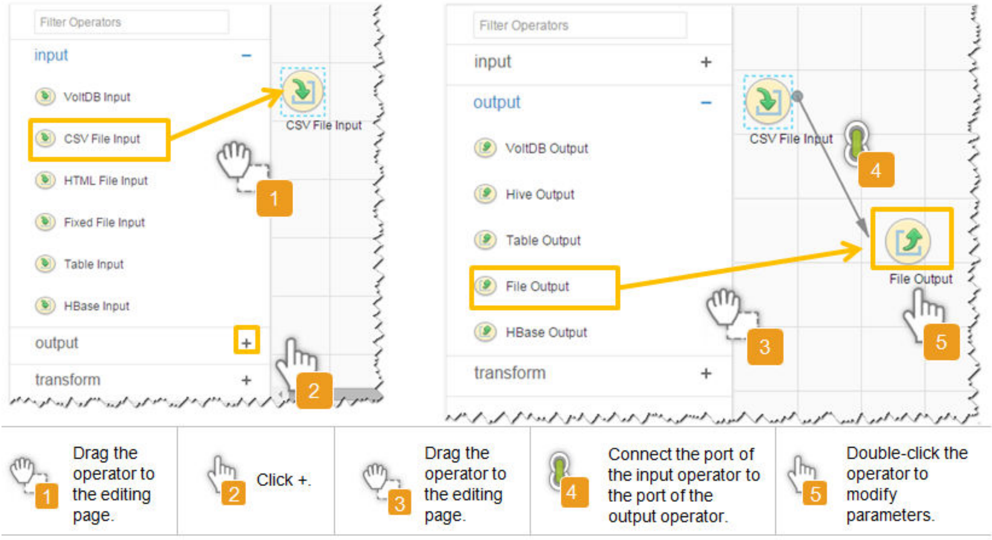
Setting Data Transformation

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 **Operator Help** and **Table 17-63**.

Table 17-63 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 17-34 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 17-64 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

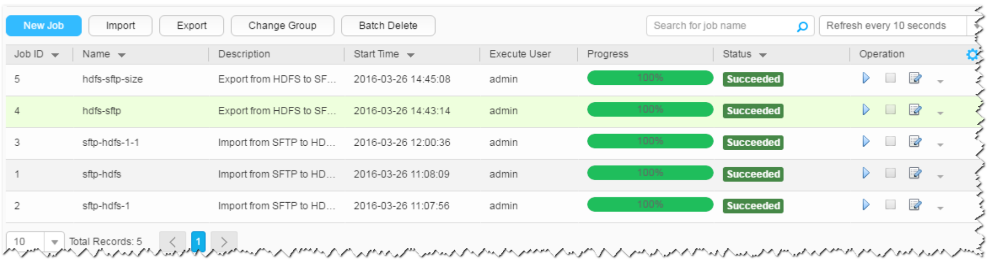
Storage Type	Applicable Scenario	Parameter	Description	Example Value
		Extractor Size	HBase does not support this parameter. Please set **Extractors**.	-
HBASE_PUTLIST	Small data volume	HBase Instance	HBase service instance that Loader selects from all available HBase service instances in the cluster. 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 17-35 Viewing job details



----End

17.12.11 Typical Scenario: Importing 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: **${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib**.
  b. Obtain the **clickhouse-jdbc-*.jar** file from the ClickHouse installation directory and save it in the **${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/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_8.1.0.1/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.

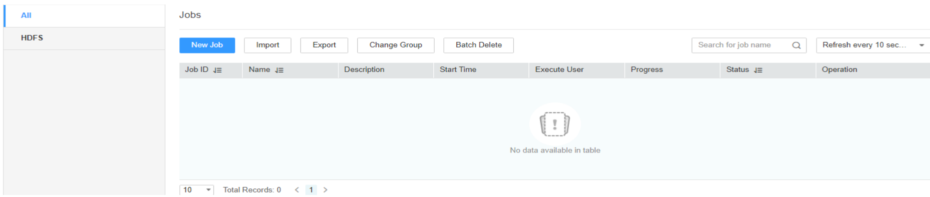
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-36 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-37 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 17-65 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 17-66 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

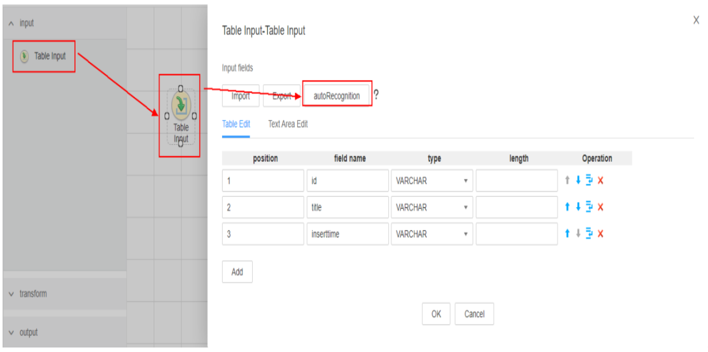
Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Operator Help** and **Table 17-67**.

Table 17-67 Input and output parameters of the operator

Input Type	Output Type
MySQL	ClickHouse

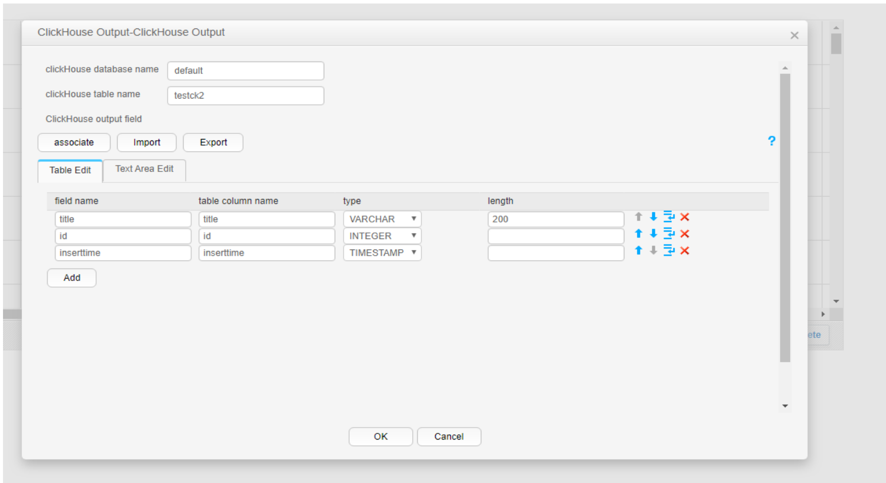
Drag **Table Input** to the grid, double-click **Table Input**, and select **autoRecognition**, as shown in **Figure 17-38**.

Figure 17-38 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 17-39**.

Figure 17-39 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 17-68 Parameter description

Parameter	Description	Example
Storage type	Select **CLICKHOUSE**.	-
ClickHouse instance	Select **ClickHouse**.	-

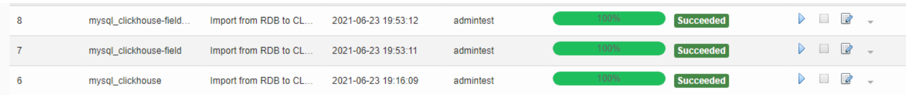
Parameter	Description	Example
Clear data before import	Select **true** or **false**. **NOTE** If you select **true** and the table to be imported is a ClickHouse distributed table, you need to manually delete the data from the local table corresponding to the ClickHouse distributed table before your 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 17-40 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

17.12.12 Typical Scenario: Importing 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.

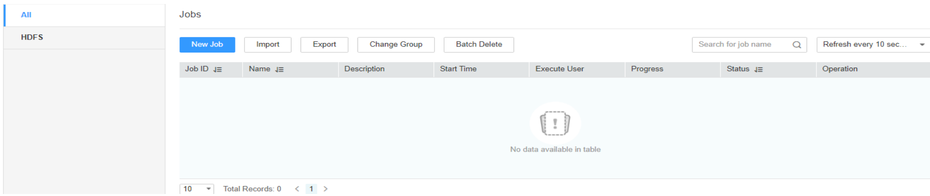
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-41 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-42 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 17-69 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 **Operator Help** and **Table 17-70**.

Table 17-70 Input and output parameters of the operator

Input Type	Output Type
CSV File Input	ClickHouse Output

Figure 17-43 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 17-71 Parameter description

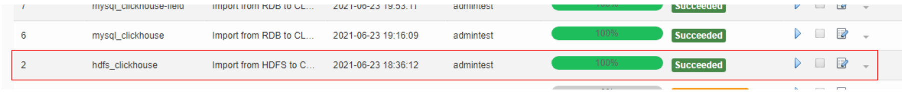
Storage Type	Parameter	Description	Example
CLICKHOUSE	ClickHouse instance	ClickHouse service instance that Loader selects from all available ClickHouse service instances in the cluster. If the selected ClickHouse service instance is not added to the cluster, the ClickHouse job cannot be run properly.	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 17-44 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

17.13 Exporting Data

17.13.1 Overview

The section "Exporting Data" applies to MRS 3.*x* or later.

Description

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.

NOTE

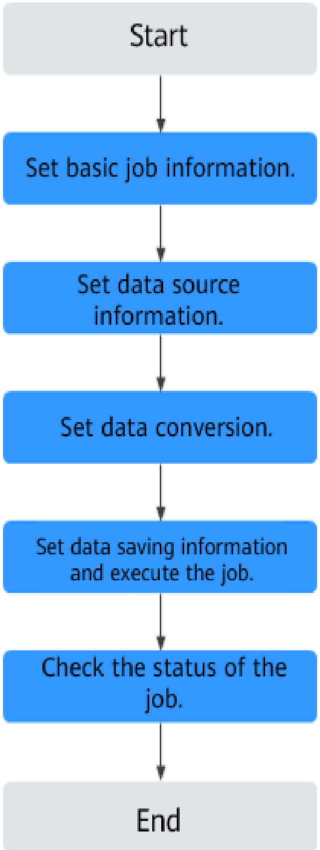
- You are advised to deploy the SFTP server, database server, and Loader into separate subnets to ensure secure data export.
- For connection to relational databases, general database connectors (generic-jdbc-connector) or dedicated database connectors (oracle-connector, oracle-partition-connector, and mysql-fastpath-connector) are available. However, compared with general database connectors, dedicated database connectors perform better in data import and export because it is 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.
- 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, dba_extents, dba_tab_partitions 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.

Export Process

A data export job can be executed in the Loader WebUI. **Figure 17-45** shows the export process.

Figure 17-45 Export process



Loader jobs can also be updated and executed using shell scripts. In this mode, the Loader client that has been installed needs to be configured.

17.13.2 Using Loader to Export Data

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_8.1.0.1/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_8.1.0.1/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

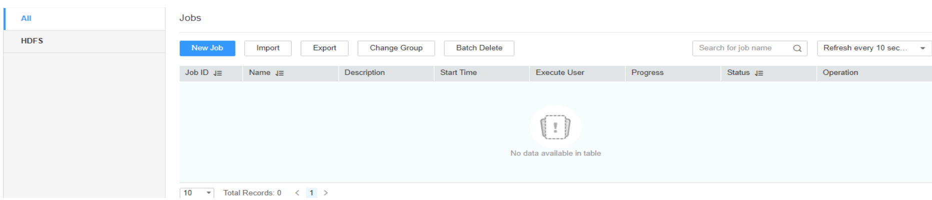
   chown omm:wheel *JAR package name*

   chmod 600 *JAR package name*
3. Log in to FusionInsight Manager. Choose **Cluster** > *Name of the desired 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 (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-46 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 17-72 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.

Connector Type	Parameter	Description
	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
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 17-73 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.

Source File Type	Parameter	Description
	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**.
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 17-74 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 **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 17-75 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.

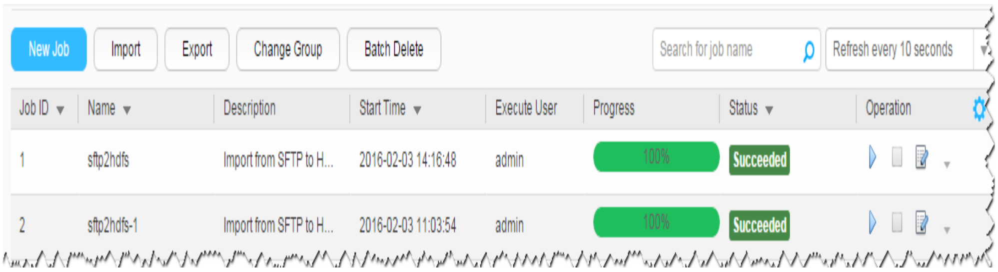
Data Connection Type	Parameter	Description
	Operation	Specifies the action during data import. When all data is to be imported from the input path to the destination path, the data is stored in a temporary directory and then copied from the temporary directory to the destination path. After the data 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 17-47 Viewing Loader Jobs



----End

17.13.3 Typical Scenario: Exporting 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.

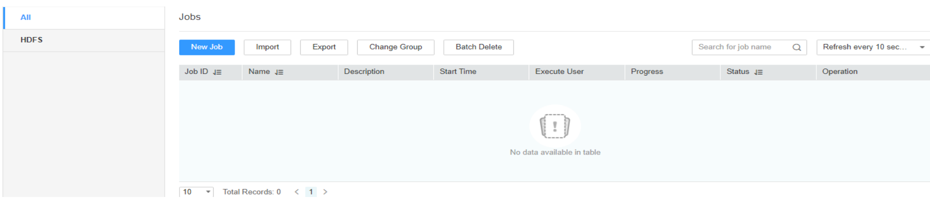
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-48 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-49 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 17-76 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 17-77 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.	-

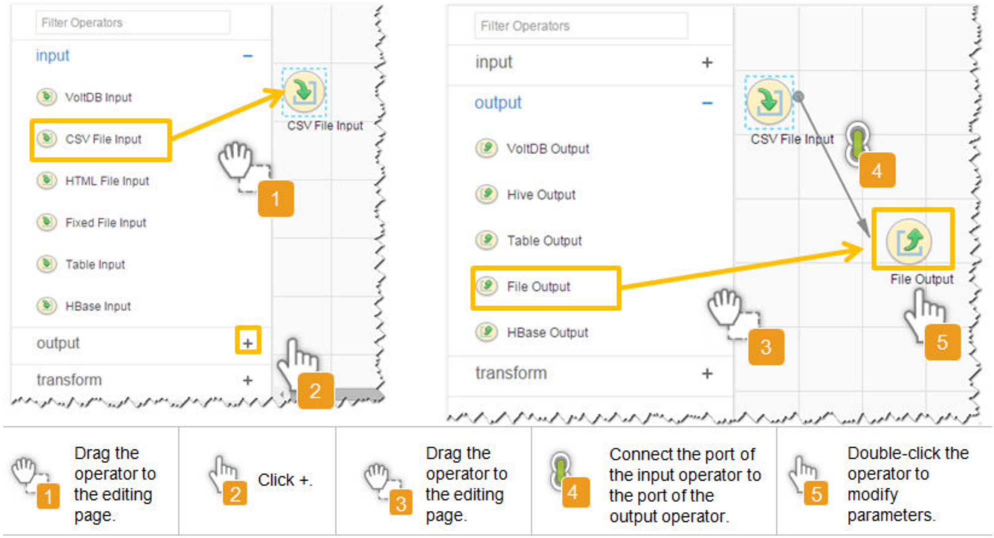
Setting Data Transformation

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 **Operator Help** and **Table 17-78**.

Table 17-78 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 17-50 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 17-79 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

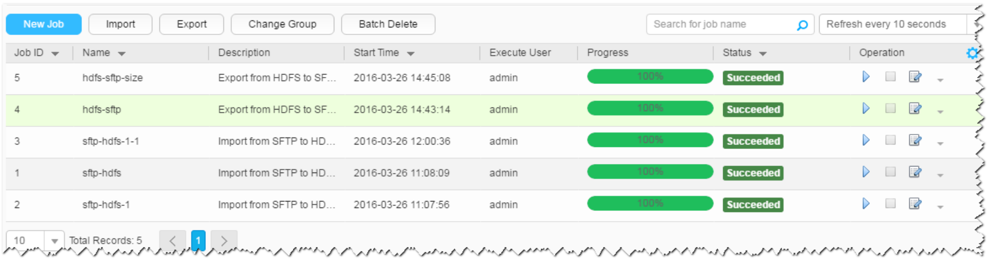
Parameter	Description	Example Value
Compression	Indicates whether to enable the compressed transmission function when SFTP is used to export data. ● The value **true** indicates that compression is enabled. ● The value **false** indicates that compression is disabled.	true

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 17-51 Viewing a job



----End

17.13.4 Typical Scenario: Exporting 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.

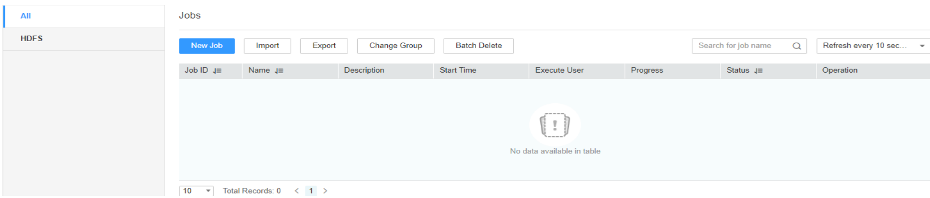
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-52 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-53 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 17-80 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 17-81 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

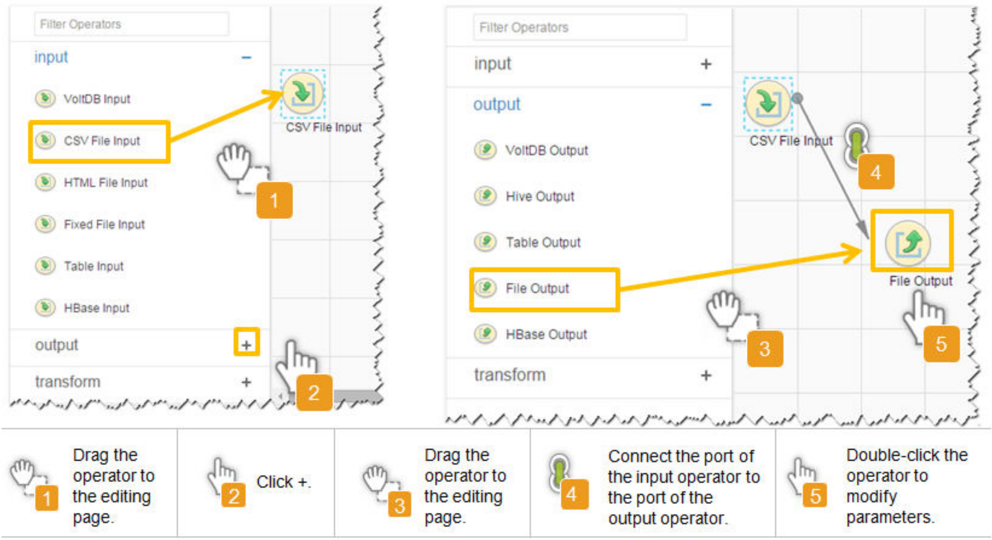
Setting Data Transformation

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 **Operator Help** and **Table 17-82**.

Table 17-82 Setting the input and output parameters of the operator

Input Type	Export Type
HBase input	File output

Figure 17-54 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 17-83 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

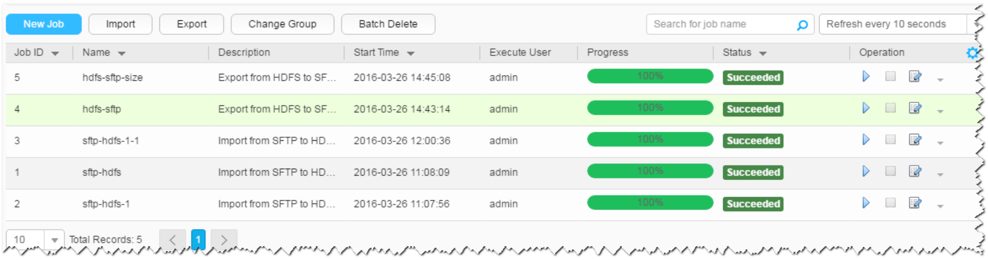
Parameter	Description	Example Value
Operation	Specifies the action during data import. When all data is to be imported from the input path to the destination path, the data is stored in a temporary directory and then copied from the temporary directory to the destination path. After the data is imported 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 17-55 Viewing a job



----End

17.13.5 Typical Scenario: Exporting 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.

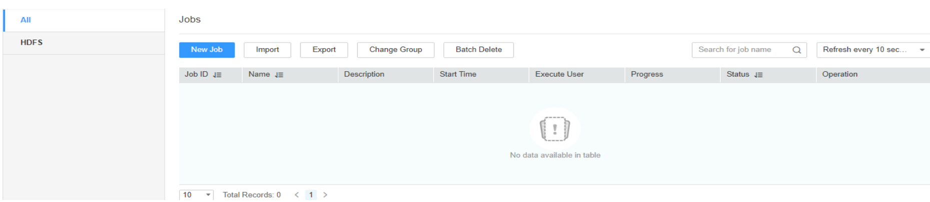
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name*, **Active)**. The Loader web UI is displayed.

Figure 17-56 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-57 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 17-84 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 17-85 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

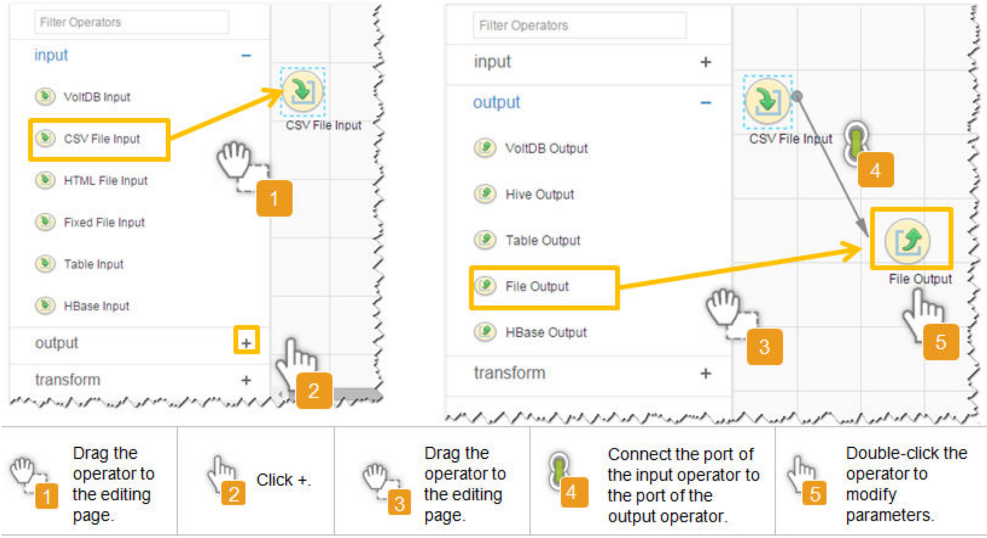
Setting Data Transformation

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 **Operator Help** and **Table 17-86**.

Table 17-86 Setting the input and output parameters of the operator

Input Type	Export Type
Hive input	File output

Figure 17-58 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 17-87 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

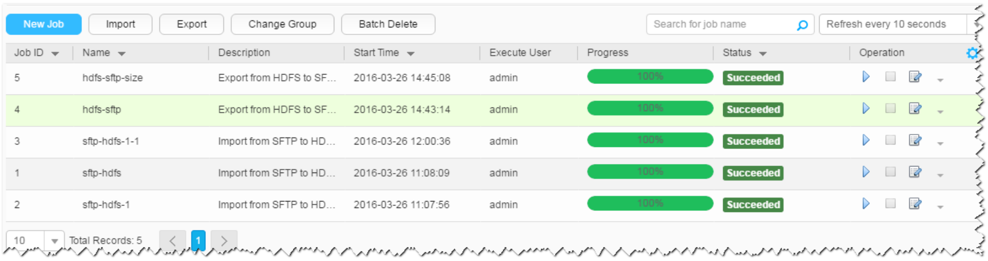
Parameter	Description	Example Value
Operation	Specifies the action during data import. When all data is to be imported from the input path to the destination path, the data is stored in a temporary directory and then copied from the temporary directory to the destination path. After the data is imported 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 17-59 Viewing a job



----End

17.13.6 Typical Scenario: Exporting 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_8.1.0.1/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_8.1.0.1/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** > *Name of the desired cluster* > **Service** > **Loader** > **More** > **Restart**. Enter the password of the administrator to restart the Loader service.

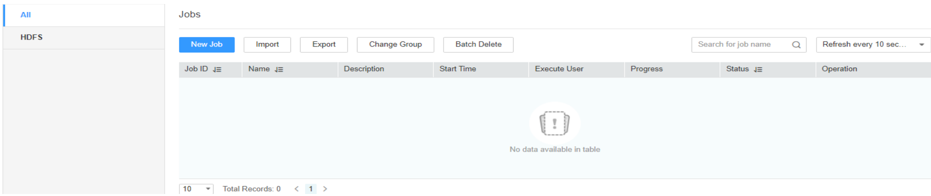
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-60 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-61 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 17-88 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 17-89 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

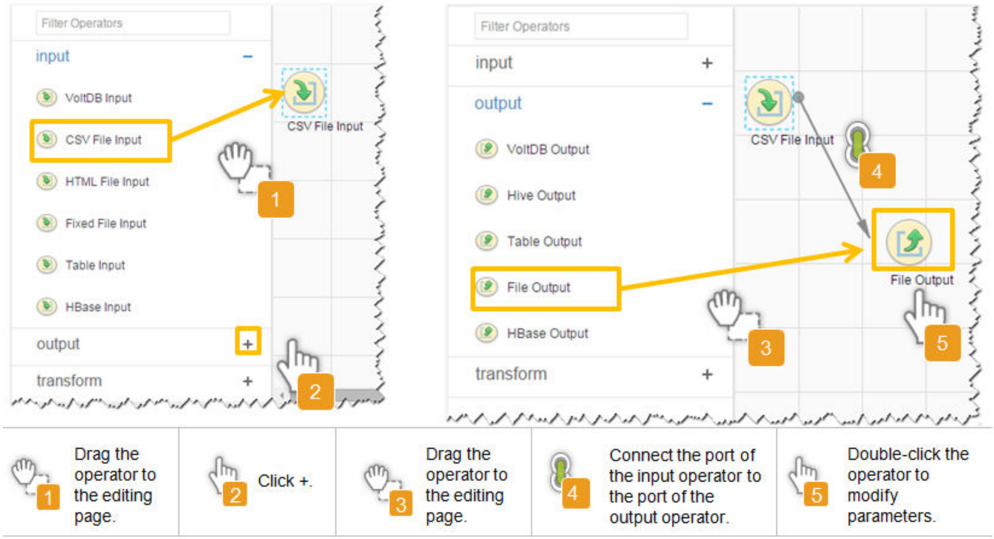
Step 5 Click **Next**. On the displayed **Transform** page, set the transformation operations in the data transformation process. For details about how to select operators and set parameters, see **Operator Help** and **Table 17-90**.

Table 17-90 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 17-62 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 17-91 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

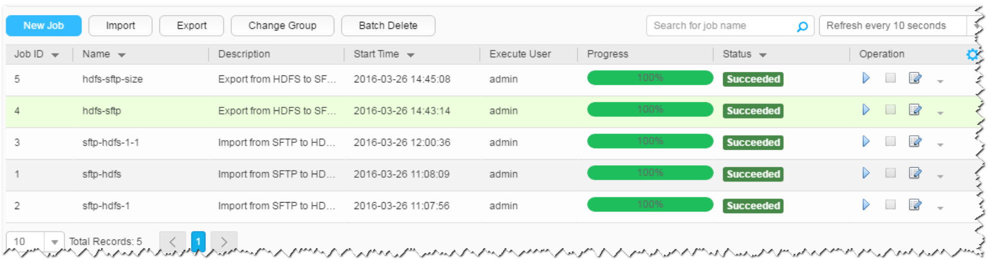
Parameter	Description	Example Value
Temporary table	Specifies the name of a temporary database table that is used to save temporary data during the transmission. The fields in the table must be the same as those in the database specified by **Table name**. **NOTE** A temporary table is used to prevent 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 17-63 Viewing a job



----End

17.13.7 Typical Scenario: Exporting 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_8.1.0.1/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_8.1.0.1/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** > *Name of the desired cluster* > **Service** > **Loader** > **More** > **Restart**. Enter the password of the administrator to restart the Loader service.

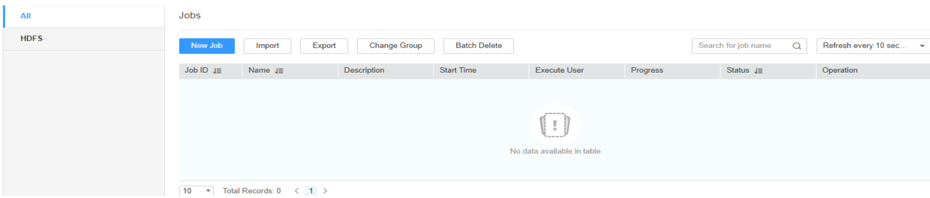
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-64 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-65 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 17-92 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

Parameter	Description	Example Value
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 17-93 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

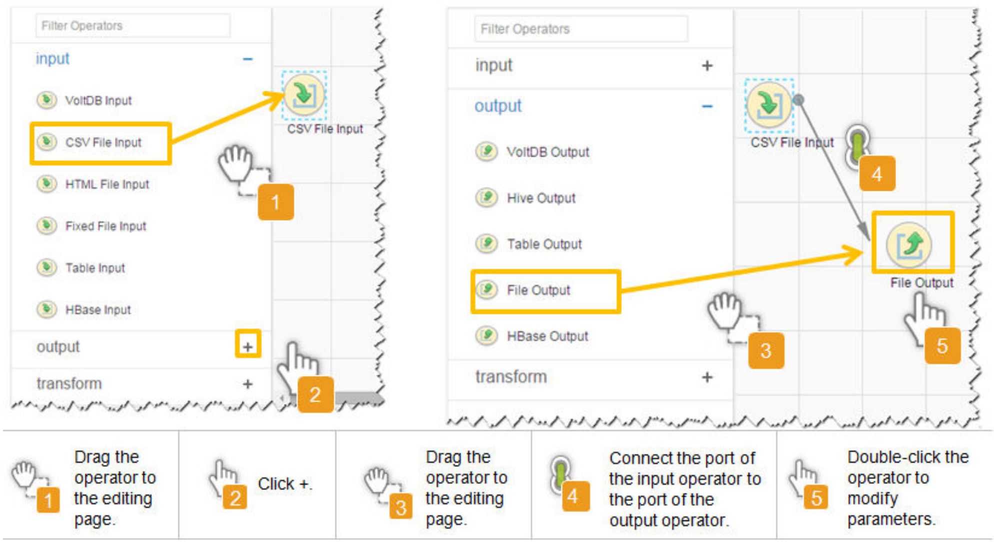
Setting Data Transformation

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 **Operator Help** and **Table 17-94**.

Table 17-94 Setting the input and output parameters of the operator

Input Type	Export Type
HBase input	Table output

Figure 17-66 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 17-95 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

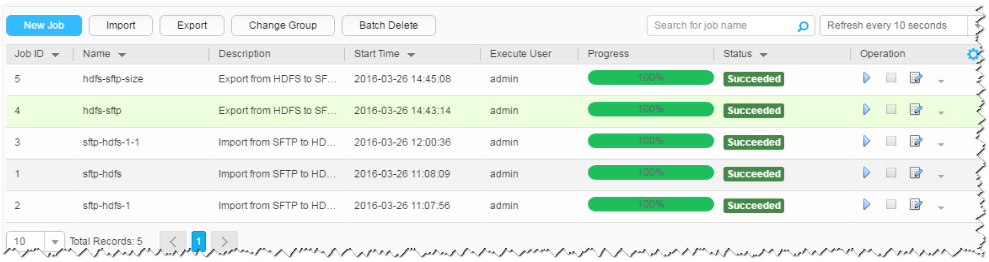
Parameter	Description	Example Value
Temporary table	Specifies the name of a temporary database table that is used to save temporary data during the transmission. The fields in the table must be the same as those in the database specified by **Table name**. **NOTE** A temporary table is used to prevent 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 17-67 Viewing a job



----End

17.13.8 Typical Scenario: Exporting 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_8.1.0.1/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_8.1.0.1/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** > *Name of the desired cluster* > **Service** > **Loader** > **More** > **Restart**. Enter the password of the administrator to restart the Loader service.

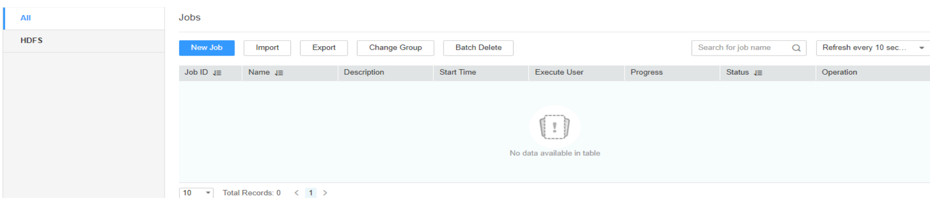
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-68 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-69 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 17-96 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

Parameter	Description	Example Value
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 17-97 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

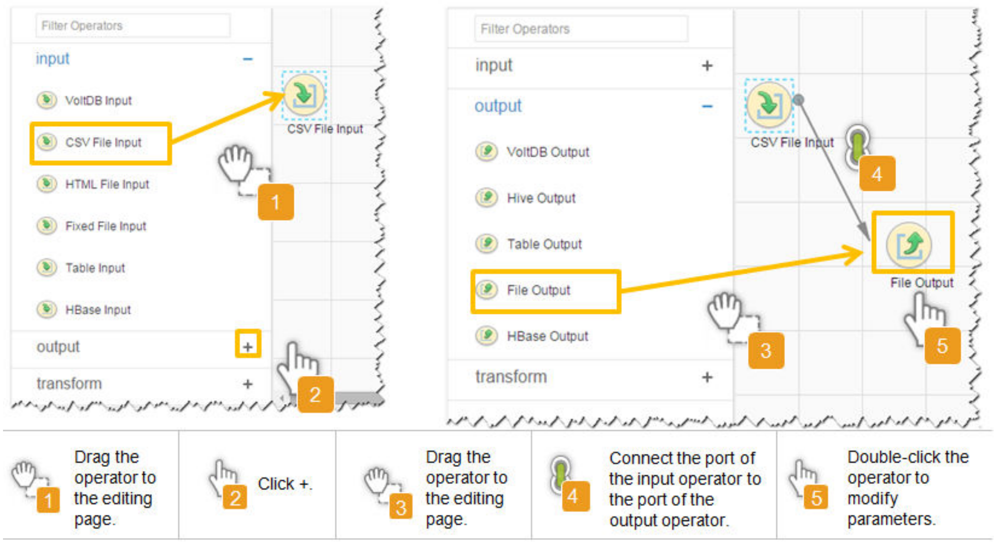
Setting Data Transformation

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 **Operator Help** and **Table 17-98**.

Table 17-98 Setting the input and output parameters of the operator

Input Type	Export Type
Hive input	Table output

Figure 17-70 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 17-99 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

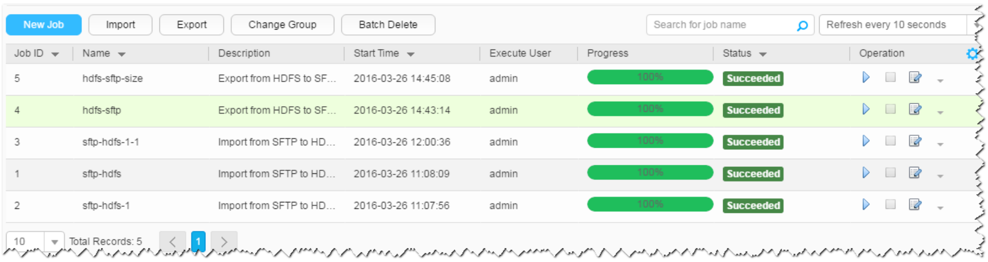
Parameter	Description	Example Value
Temporary table	Specifies the name of a temporary database table that is used to save temporary data during the transmission. The fields in the table must be the same as those in the database specified by **Table name**. **NOTE** A temporary table is used to prevent 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 17-71 Viewing a job



----End

17.13.9 Typical Scenario: Importing 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.

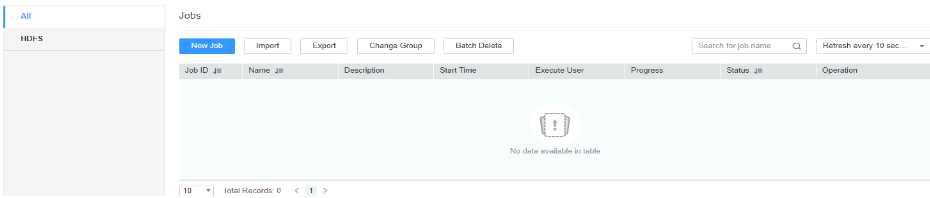
Procedure

Setting Basic Job Information

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-72 Loader web UI



Step 2 Click **New Job** to go to the **Basic Information** page and set basic job information.

Figure 17-73 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 17-100 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

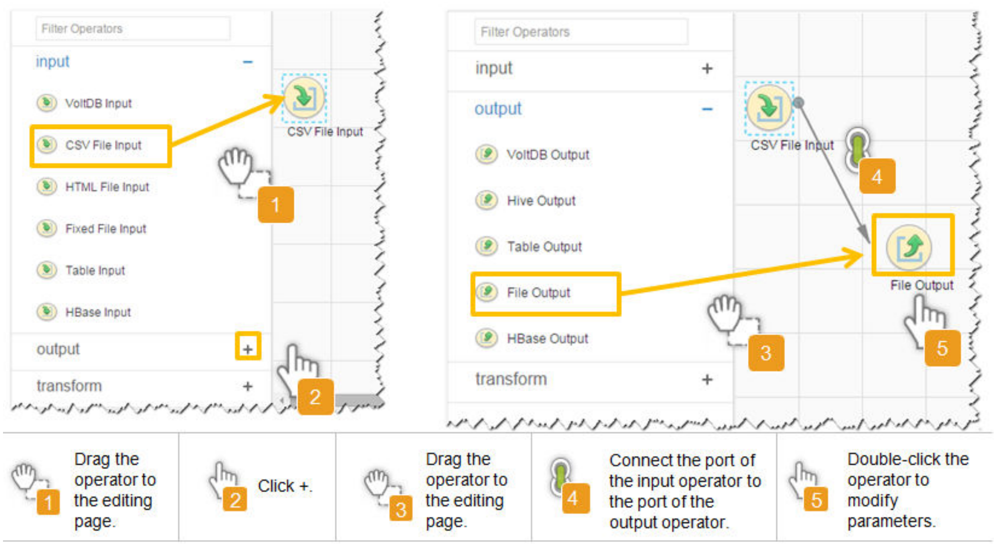
Setting Data Transformation

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 **Operator Help** and **Table 17-101**.

Table 17-101 Setting the input and output parameters of the operator

Input Type	Export Type
HBase input	File output

Figure 17-74 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 17-102 Parameter description

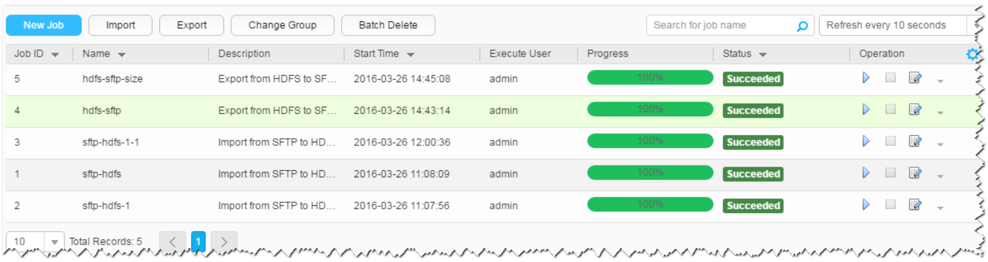
Parameter	Description	Example
Output path	Specifies the output directory or file name of the export file in the HDFS or OBS. **NOTE** You can use macros to define path parameters. For details, see **Using Macro Definitions in Configuration Items** .	/user/test
File Format	Specifies the file export type. ● **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 17-75 Viewing a job



----End

17.14 Managing Jobs

17.14.1 Migrating Loader Jobs in Batches

Scenario

Loader allows jobs to be migrated in batches from a group (source group) to another group (target group).

This section applies to MRS 3.x or later.

Prerequisites

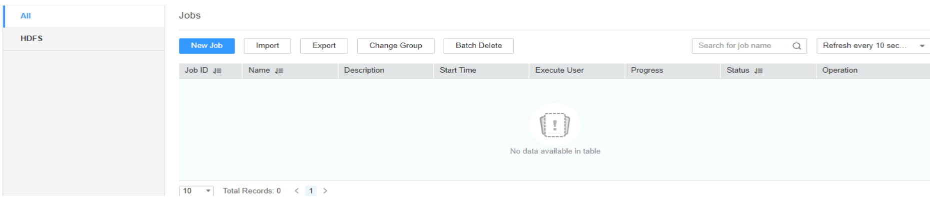
- The source group and target group exist.
- The current user has the **Group Edit** permission for the source group and target group.
- The current user has the **Jobs Edit** permission for the source group or the **Edit** permission for the jobs to be migrated.

Procedure

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-76 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

17.14.2 Deleting Loader Jobs in Batches

Scenario

Loader allows existing jobs to be deleted in batches.

This section applies to MRS 3.x or later.

Prerequisites

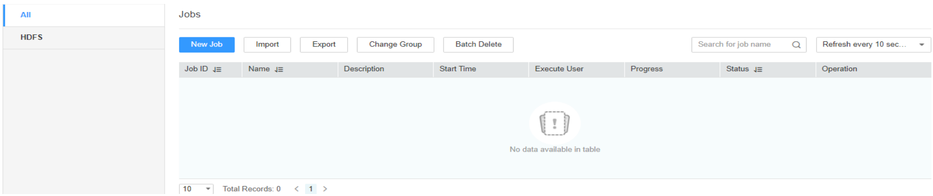
The current user has the **Edit** permission for the jobs to be deleted or the **Jobs Edit** permission for the group to which the jobs belong.

Procedure

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-77 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

17.14.3 Importing Loader Jobs in Batches

Scenario

Loader allows all jobs of a configuration file to be imported in batches.

This section applies to MRS 3.x or later.

Prerequisites

The current user has the **Jobs Edit** permission of the group to which the jobs to be imported belong.

NOTE

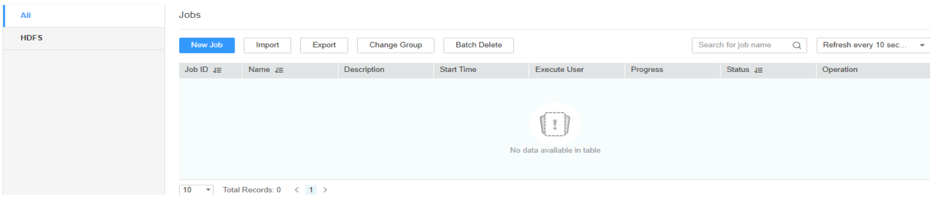
If the group to which the jobs to be imported belong does not exist, the group is automatically created first. The current user is the creator of the group and has the **Jobs Edit** permission of the group.

Procedure

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name*, **Active)**. The Loader web UI is displayed.

Figure 17-78 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

17.14.4 Exporting Loader Jobs in Batches

Scenario

Loader allows existing jobs to be exported in batches.

This section applies to MRS 3.x or later.

Prerequisites

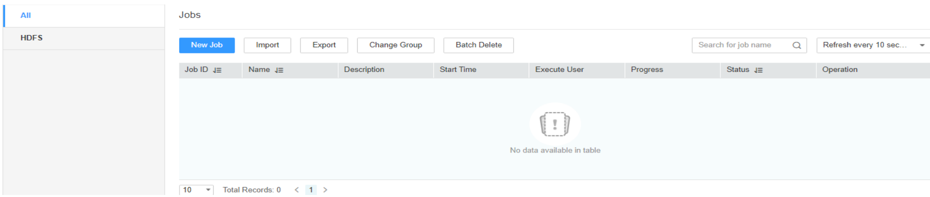
The current user has the **Edit** permission for the jobs to be exported or the **Jobs Edit** permission of the group to which the jobs belong.

Procedure

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-79 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

17.14.5 Viewing Historical Job Information

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.

This section applies to MRS 3.x or later.

Prerequisites

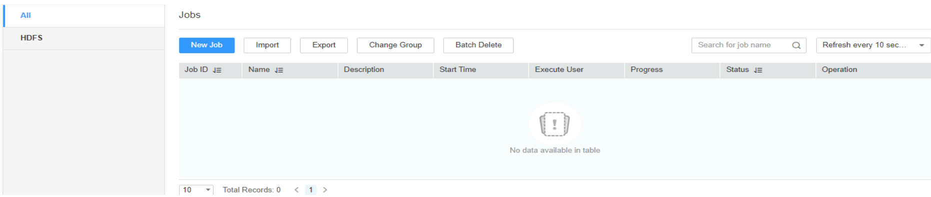
You have obtained the username and password for logging in to the Loader WebUI.

Procedure

Step 1 Access the Loader web UI.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-80 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 17-81 Viewing historical records

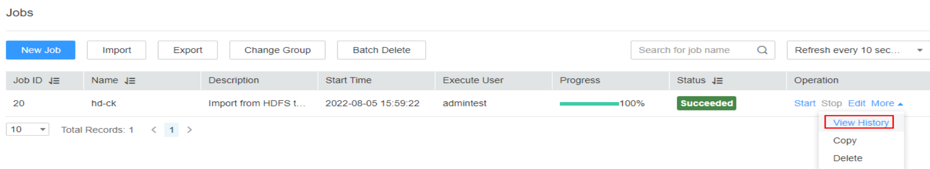


Table 17-103 Parameters

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

17.15 Operator Help

17.15.1 Overview

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 17-104 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.

17.15.2 Input Operators

17.15.2.1 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 17-105 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.

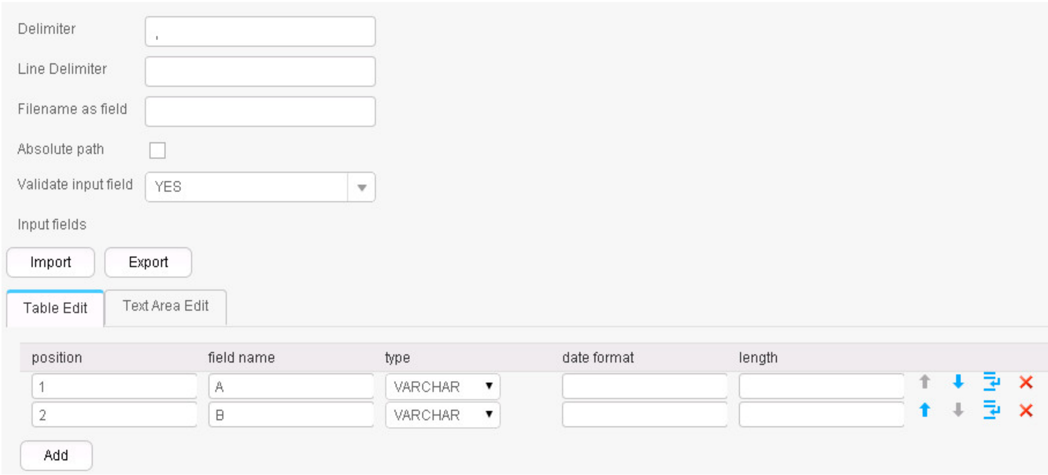
- If the number of input field columns is equal to the number of field columns actually included in the original data, the data in the line will become dirty data.

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
```

17.15.2.2 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 17-106 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

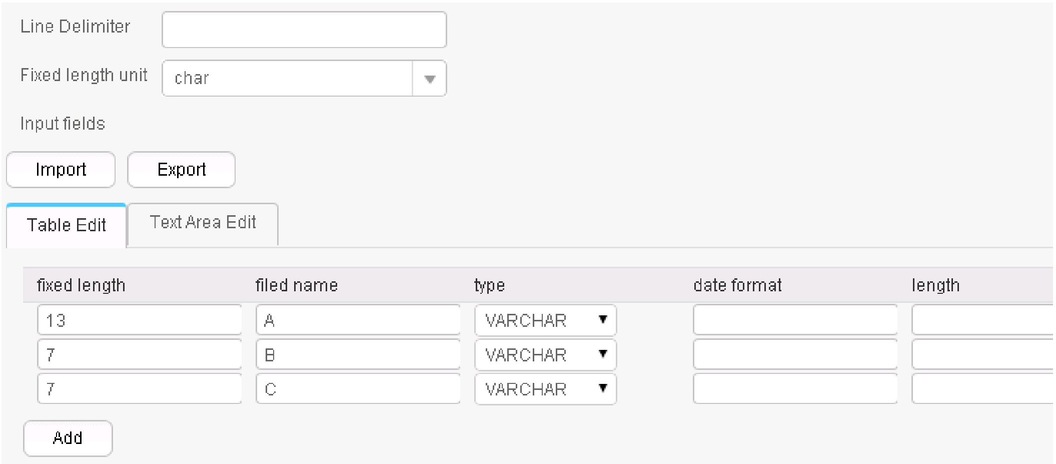
- The source file is split based on the input field length to generate fields.
- If the field value does not match the actual type, the data in the line will become dirty data.
- If the field split length is greater than the length of the original field value, the data split fails and the line becomes 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
```

17.15.2.3 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 17-107 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

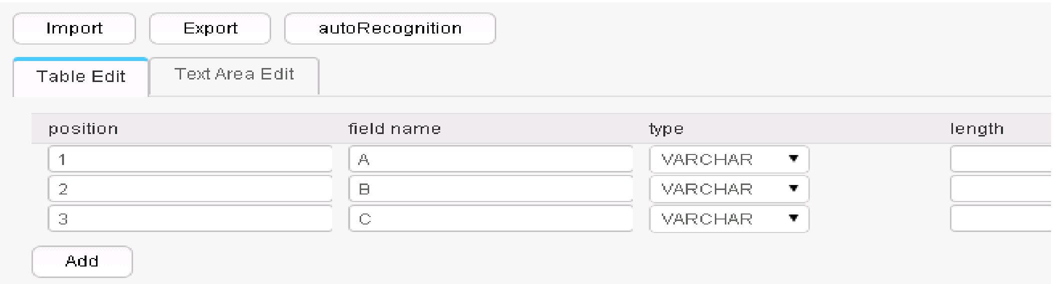
Configure the **Table Input** operator to generate the following fields:

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
```

17.15.2.4 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 17-108 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:

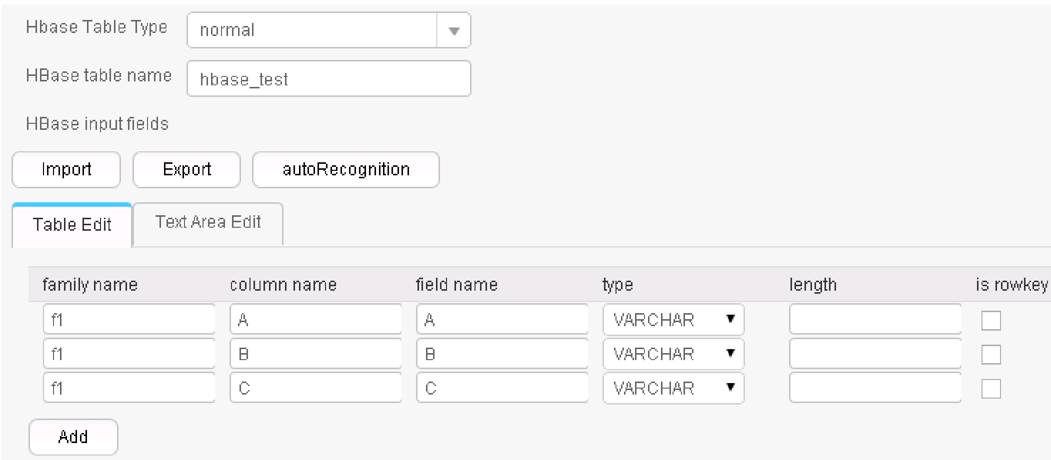
create table test_1 (id int, name text, value text);

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

17.15.2.5 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 17-109 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.

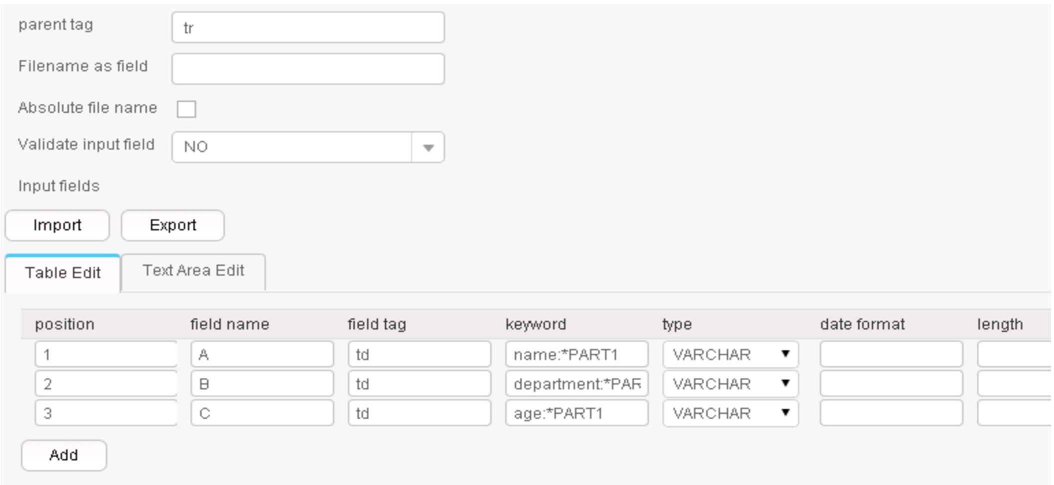
- If the tag is configured incorrectly, the obtained value is empty, but no error is reported.

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
```

17.15.2.6 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 17-110 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:

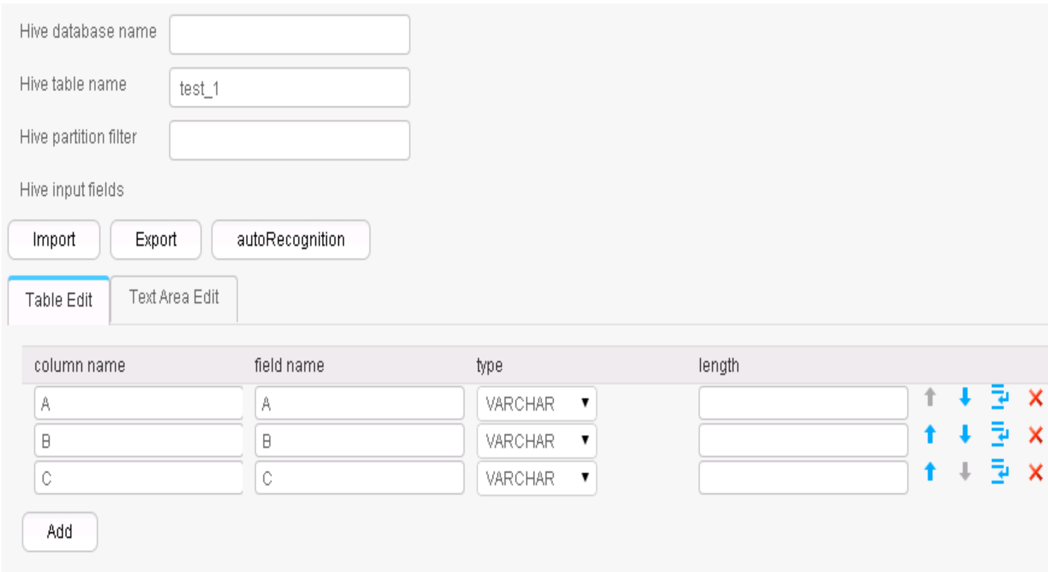
create table test_1 (id int, name text, value text);

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

17.15.2.7 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 17-111 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:

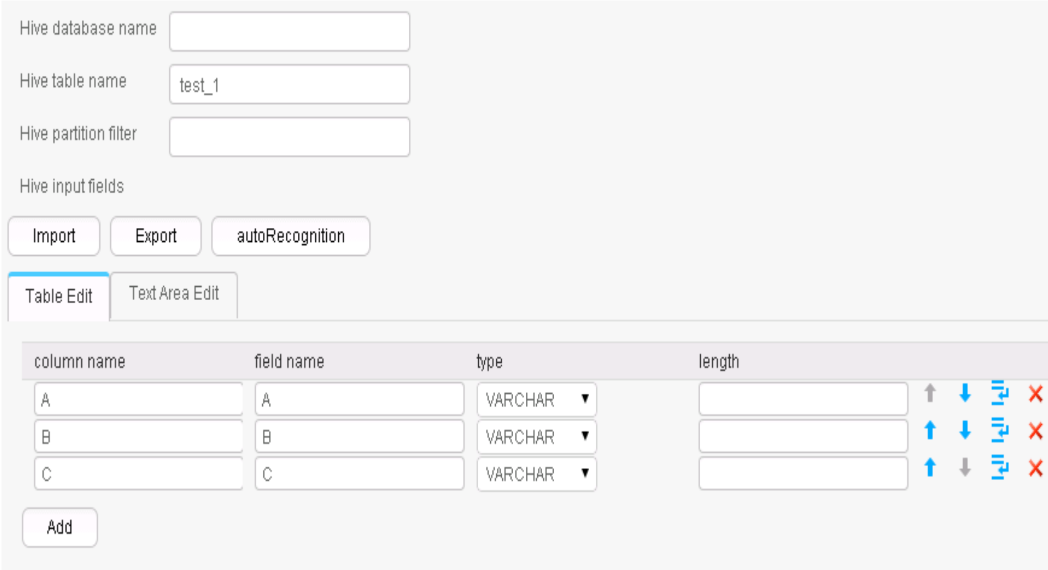
create table test_1 (id int, name text, value text);

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

17.15.3 Conversion Operators

17.15.3.1 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 17-112 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.

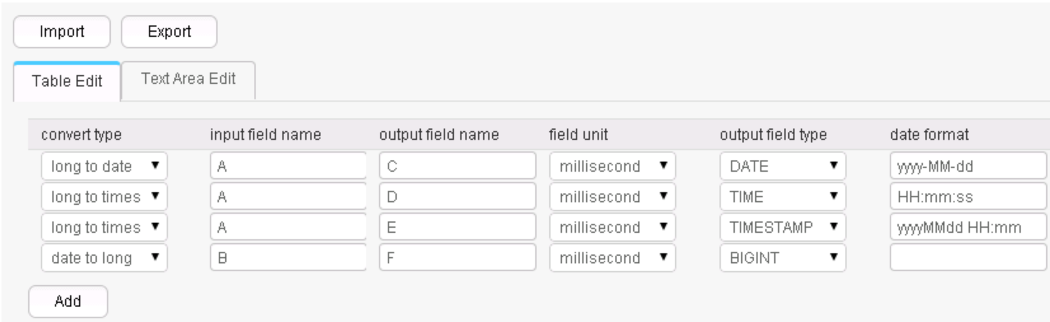
Example

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
```

17.15.3.2 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 17-113 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
```

17.15.3.3 Constant Field Addition

Overview

The **Add Constants** operator generates constant fields.

Input and Output

- Input: none
- Output: constant fields

Parameter Description

Table 17-114 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
```

17.15.3.4 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 17-115 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.

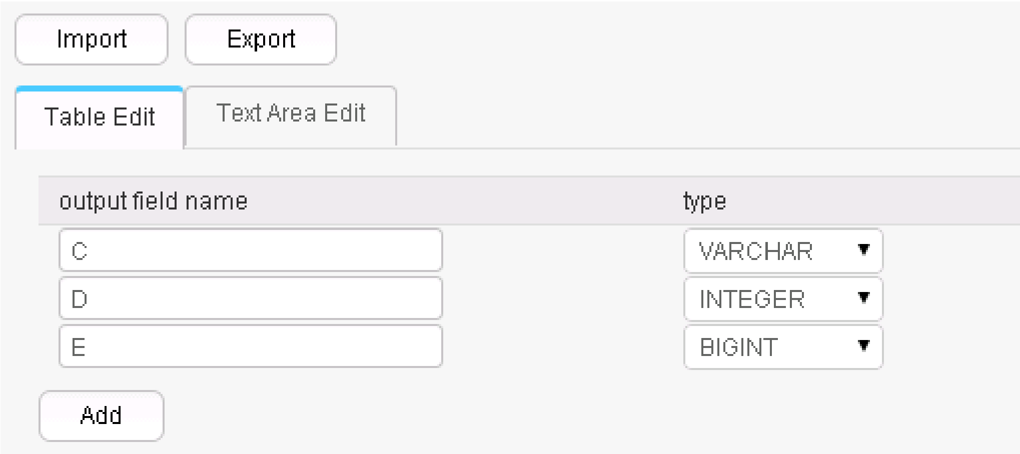
Example

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,1631427868,8679141166895017 57
key3,,2jg5e7b1m17kq,654806209,2477823020516316030
```

The random value fields generated each time are different.

17.15.3.5 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 17-116 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.

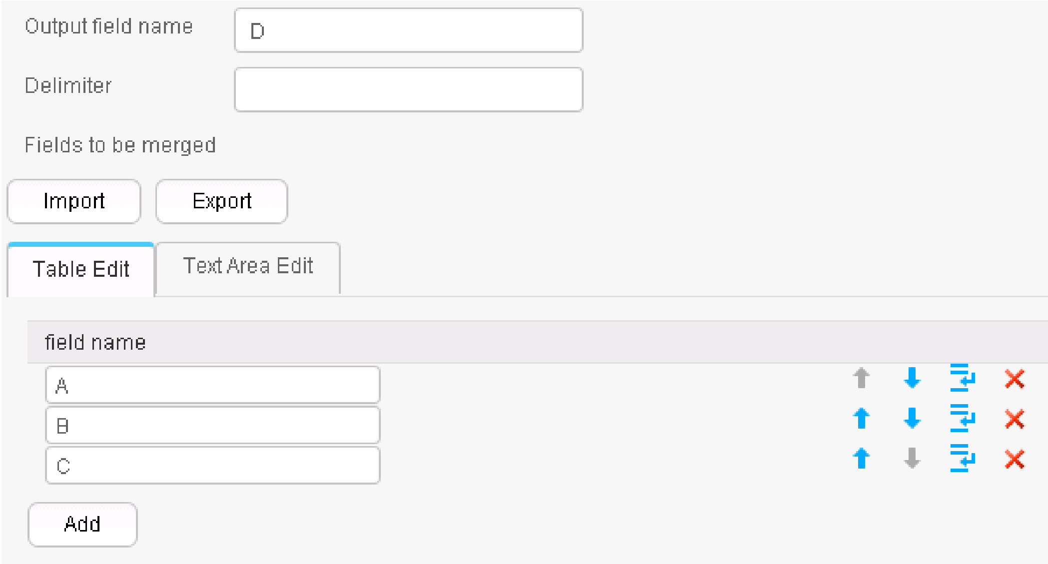
Example

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
```

17.15.3.6 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 17-117 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.

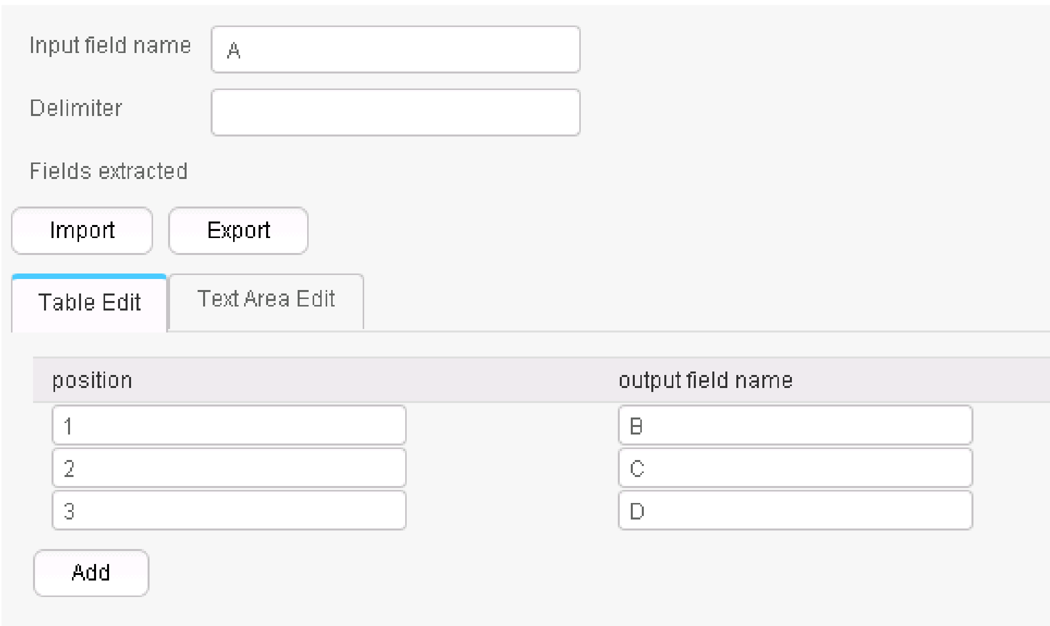
Example

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
```

17.15.3.7 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 17-118 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.

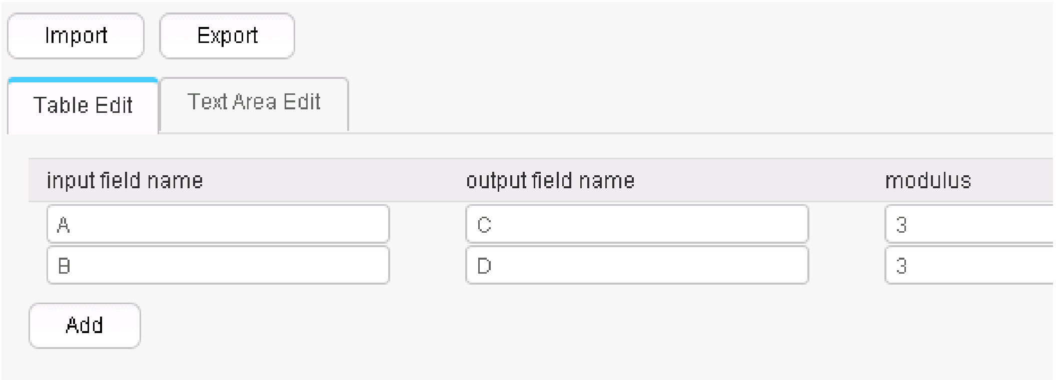
Example

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
```

17.15.3.8 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 17-119 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
```

17.15.3.9 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 17-120 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.

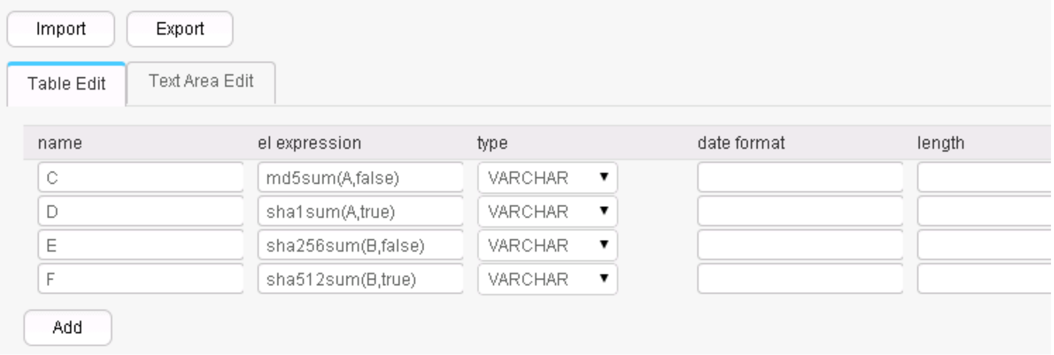
Example

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
```

17.15.3.10 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 17-121 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.

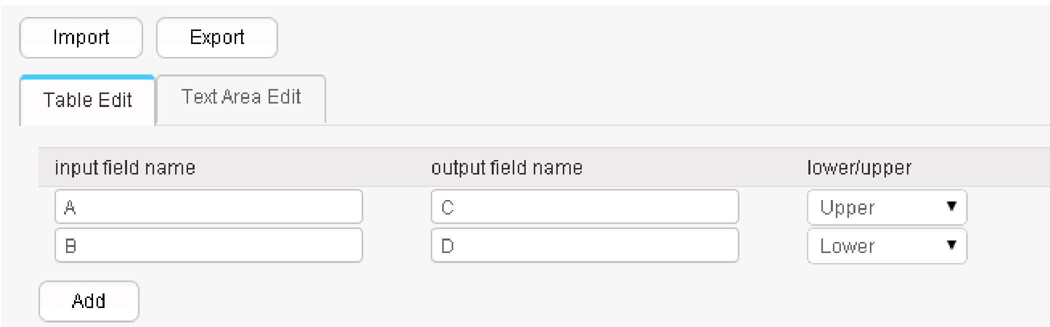
Example

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,dcba,tcudorp
FusionInsight,Bigdata,thgisnInoisuF,atadgiB
```


17.15.3.11 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 17-122 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.

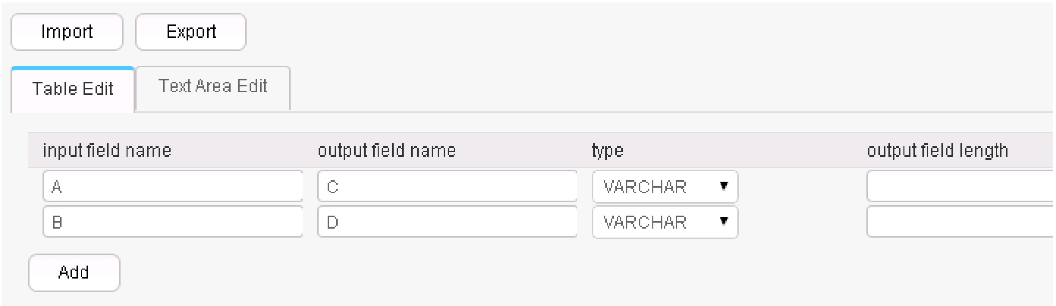
Example

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
```

17.15.3.12 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 17-123 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.

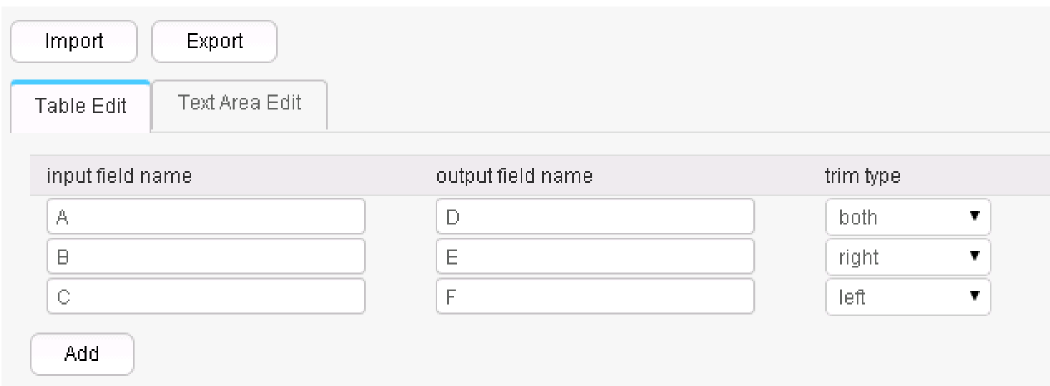
Example

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
```

17.15.3.13 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 17-124 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.

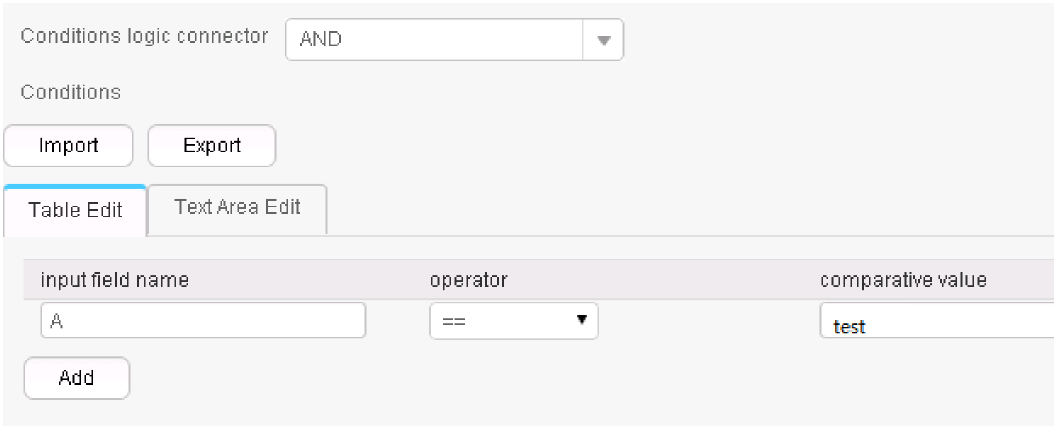
Example

Use the **CSV File Input** operator to generate two fields A and B.

The following figure shows the source file.


test, product
FusionInsight,Bigdata

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
```

17.15.3.14 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 17-125 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.

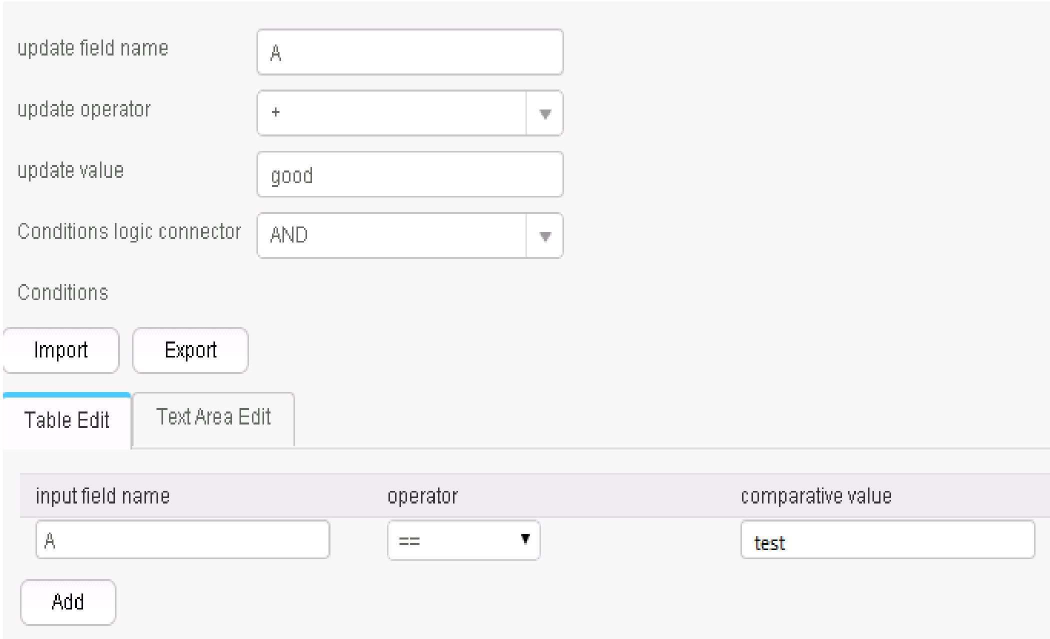
Example

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.

17.15.4 Output Operators

17.15.4.1 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 17-126 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.

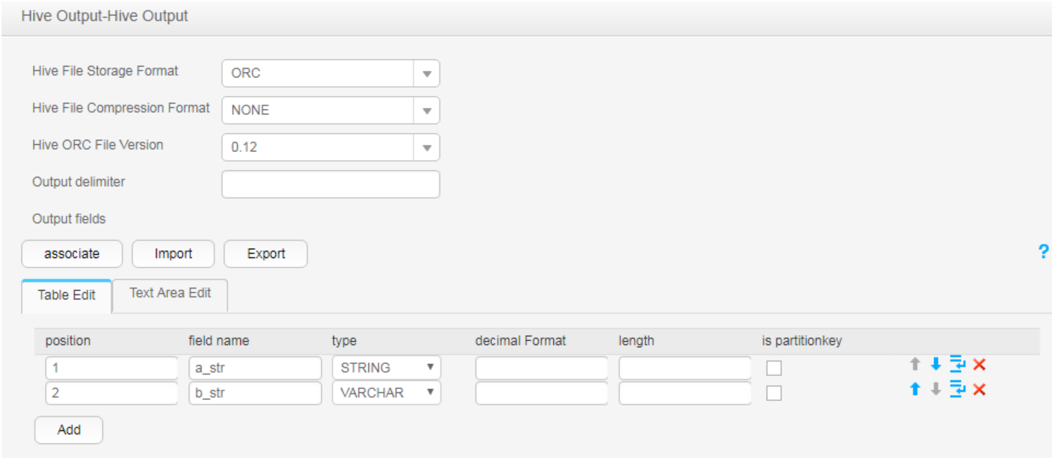
Example

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)
```

17.15.4.2 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 17-127 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.

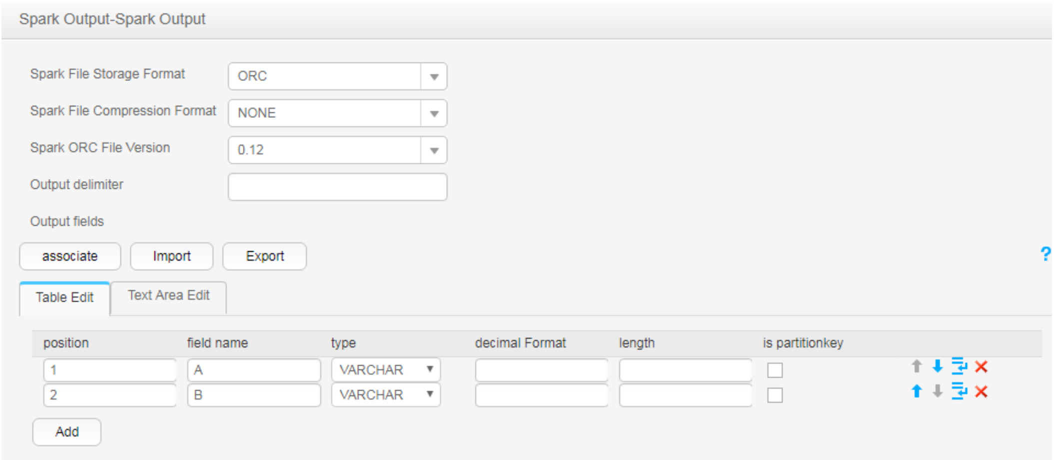
Example

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.



17.15.4.3 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 17-128 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

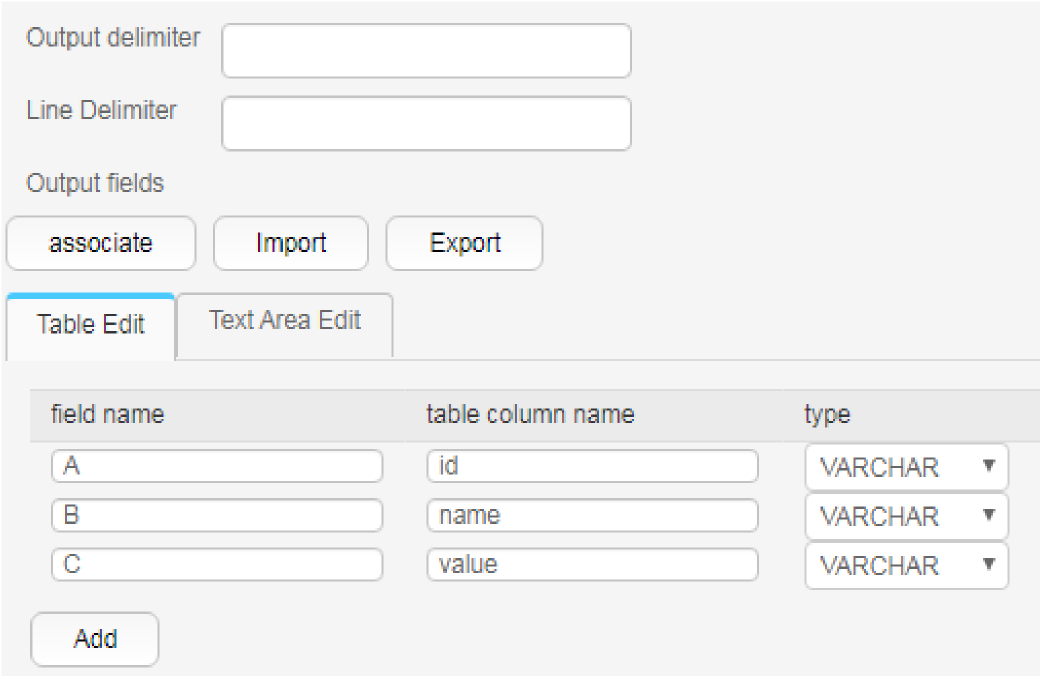
Use the data export from HBase to sqlserver2014 as an example.

In sqlserver2014, run the following statement to create an empty data test_1 for storing HBase data. Run the following statement:

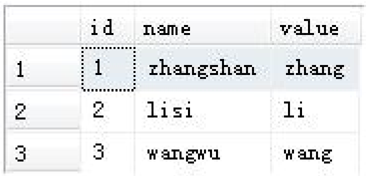
create table test_1 (id int, name text, value text);

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

17.15.4.4 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 17-129 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.

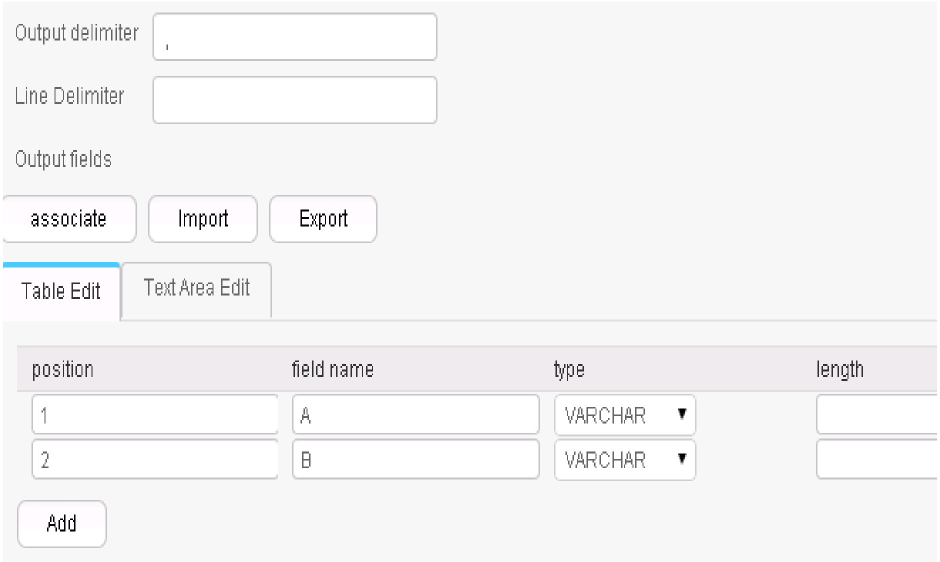
Example

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
```

17.15.4.5 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 17-130 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. ● column family name: HBase column family name. If it is not configured during HBase/Phoenix table creation, the default value is **'0'**. ● 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

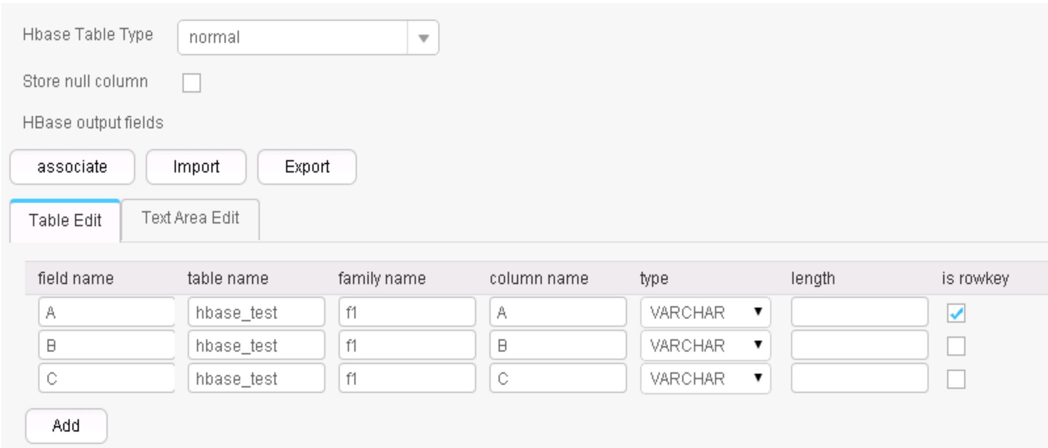
Using table input as an example, after the fields are generated, the HBase Output operator exports them to the related HBase table and stores the data in the test table, as shown in the following figure.

	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
```

17.15.4.6 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 17-131 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),

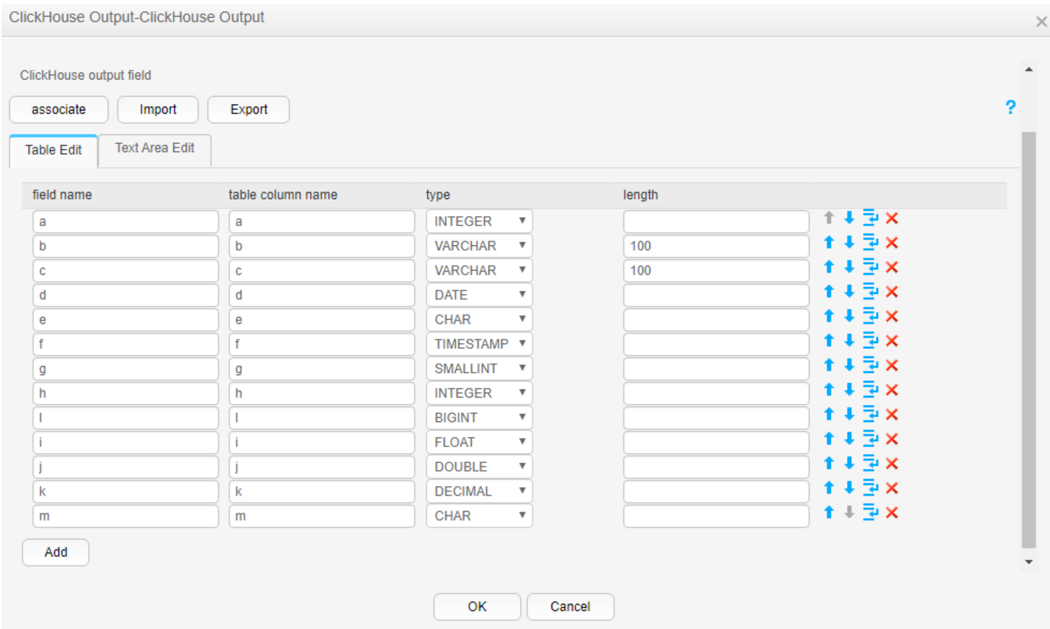
m boolean

)

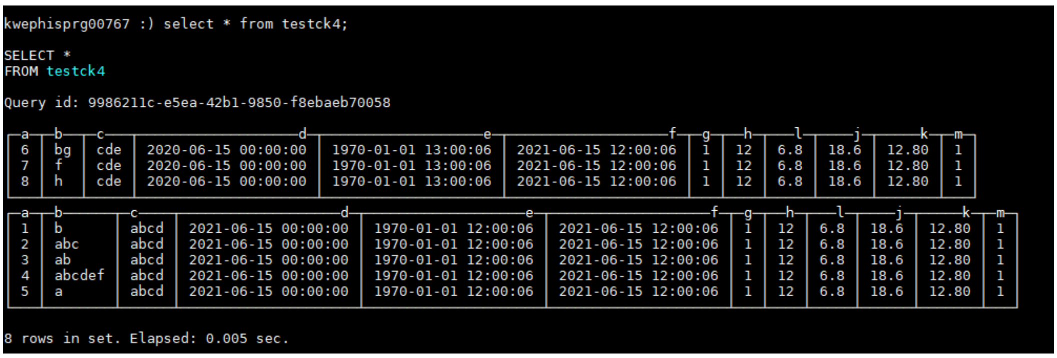
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/testck4', '{replica}')

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.



```
kwephisprg00767 :) select * from testck4;

SELECT *
FROM testck4

Query id: 9986211c-e5ea-42b1-9850-f8ebaeb70058

┌─a─┬─b──┬─c───┬───────────────────d─┬───────────────────e─┬───────────────────f─┬─g─┬──h─┬───l─┬────j─┬─────k─┬─m─┐
│ 6 │ bg │ cde │ 2020-06-15 00:00:00 │ 1970-01-01 13:00:06 │ 2021-06-15 12:00:06 │ 1 │ 12 │ 6.8 │ 18.6 │ 12.80 │ 1 │
│ 7 │ f  │ cde │ 2020-06-15 00:00:00 │ 1970-01-01 13:00:06 │ 2021-06-15 12:00:06 │ 1 │ 12 │ 6.8 │ 18.6 │ 12.80 │ 1 │
│ 8 │ h  │ cde │ 2020-06-15 00:00:00 │ 1970-01-01 13:00:06 │ 2021-06-15 12:00:06 │ 1 │ 12 │ 6.8 │ 18.6 │ 12.80 │ 1 │
└───┴────┴─────┴─────────────────────┴─────────────────────┴─────────────────────┴───┴────┴─────┴──────┴───────┴───┘
┌─a─┬─b──────┬─c────┬───────────────────d─┬───────────────────e─┬───────────────────f─┬─g─┬──h─┬───l─┬────j─┬─────k─┬─m─┐
│ 1 │ b      │ abcd │ 2021-06-15 00:00:00 │ 1970-01-01 12:00:06 │ 2021-06-15 12:00:06 │ 1 │ 12 │ 6.8 │ 18.6 │ 12.80 │ 1 │
│ 2 │ abc    │ abcd │ 2021-06-15 00:00:00 │ 1970-01-01 12:00:06 │ 2021-06-15 12:00:06 │ 1 │ 12 │ 6.8 │ 18.6 │ 12.80 │ 1 │
│ 3 │ ab     │ abcd │ 2021-06-15 00:00:00 │ 1970-01-01 12:00:06 │ 2021-06-15 12:00:06 │ 1 │ 12 │ 6.8 │ 18.6 │ 12.80 │ 1 │
│ 4 │ abcdef │ abcd │ 2021-06-15 00:00:00 │ 1970-01-01 12:00:06 │ 2021-06-15 12:00:06 │ 1 │ 12 │ 6.8 │ 18.6 │ 12.80 │ 1 │
│ 5 │ a      │ abcd │ 2021-06-15 00:00:00 │ 1970-01-01 12:00:06 │ 2021-06-15 12:00:06 │ 1 │ 12 │ 6.8 │ 18.6 │ 12.80 │ 1 │
└───┴────────┴──────┴─────────────────────┴─────────────────────┴─────────────────────┴───┴────┴─────┴──────┴───────┴───┘

8 rows in set. Elapsed: 0.005 sec.
```

17.15.5 Associating, Editing, Importing, or Exporting the Field Configuration of an Operator

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.

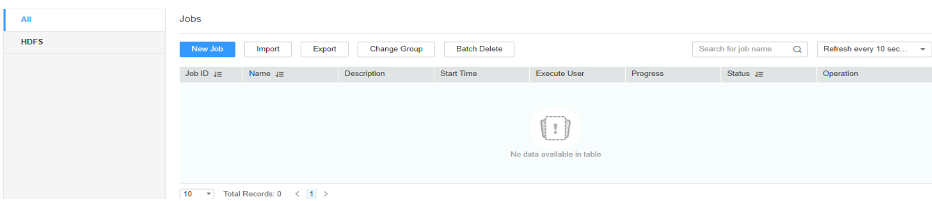
Procedure

- **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 (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-82 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.

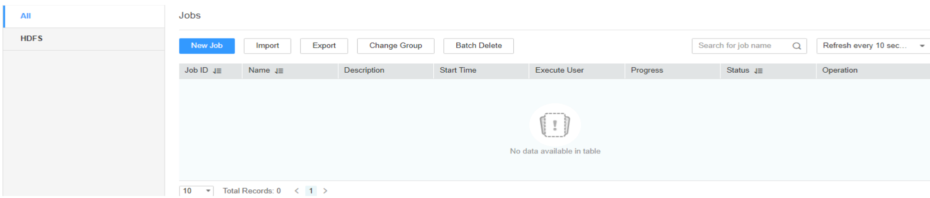
----End

- **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 (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-83 Loader web UI

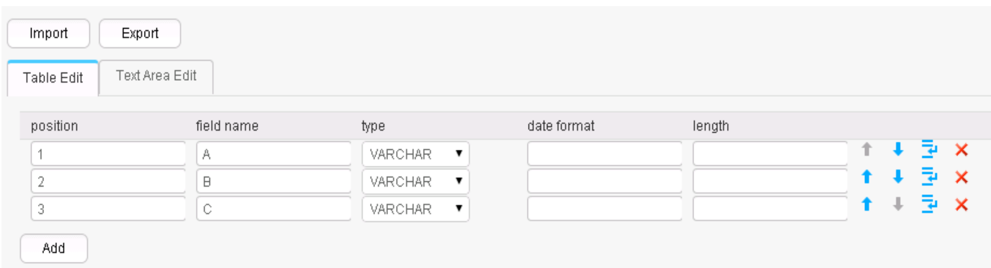


Step 2 Edit an existing job or create a new job. The **Transform** page is displayed.

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**.

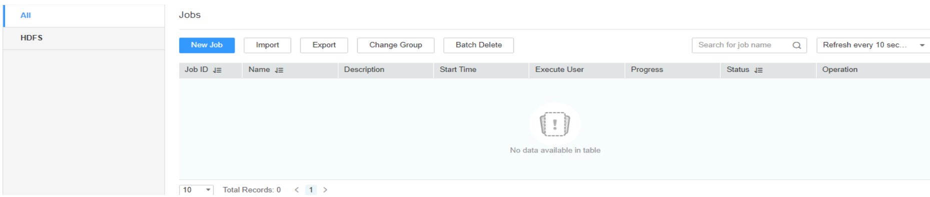
----End

- **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 (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-84 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.

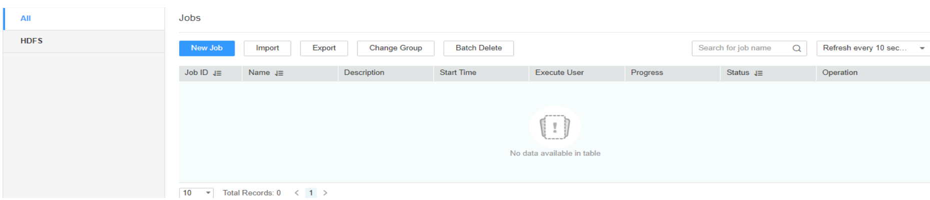
----End

- **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 (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Services** > **Loader**.
3. Click **LoaderServer(***Node name***, Active)**. The Loader web UI is displayed.

Figure 17-85 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

17.15.6 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 17-132 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**.

17.15.7 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 17-133 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.

17.16 Client Tools

17.16.1 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.

This section applies to MRS 3.*x* or later.

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 17-134**.

   vi loader-tools-1.99.3/loader-tool/job-config/login-info.xml

Table 17-134 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. **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 17-135 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

17.16.2 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.

This section applies to MRS 3.*x* or later.

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 17-136 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>Password ciphertext</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 17-137 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 17-138 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

17.16.3 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.

This section applies to MRS 3.*x* or later.

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>Password ciphertext</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

17.16.4 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.

This section applies to MRS 3.*x* or later.

Parameters

Table 17-139 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: **./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. 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.	-

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 17-140 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 17-140** 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 (,)**.

17.16.5 schedule-tool Usage Example

Scenario

After a job is created using the Loader WebUI or Loader-tool, use schedule-tool to execute the job.

This section applies to MRS 3.*x* or later.

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 17-86 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 **Typical Scenario: Importing 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**.

- 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

17.16.6 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.

This section applies to MRS 3.*x* or later.

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 17-141**.

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 17-141 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

<table>
<tr><th>Configuration parameters</th><th>Description</th><th>Example Value</th></tr>
<tr><td>authentication.password</td><td>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.
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.</td><td>-</td></tr>
</table>

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

17.16.7 Open Source sqoop-shell Tool Usage Guide

Overview

This section applies to MRS 3.*x* or later.

Sqoop-shell is an open source shell tool. 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 17-142**.

Table 17-142 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.

Command	Description
help	Views the tool help information.
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 **Example for Using the Open-Source sqoop-shell Tool (SFTP-HDFS)** and **Example for Using the Open-Source sqoop-shell Tool (Oracle-HBase)**. 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*

  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.

  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.

  1. 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*

  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.

  2. 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.
	-c,--cid	Displays information about the connection type of a specified ID.

Attribute Type	Subattribute	Description
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.
	-cn,--cname	Specifies the name of a specified connector type.
job	-x,--xid	Specifies the connector ID.

Attribute Type	Subattribute	Description
	-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.
	-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.

Attribute Type	Subattribute	Description
	-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 17-143 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.
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.

Property	Description
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

17.16.8 Example for Using the Open-Source sqoop-shell Tool (SFTP-HDFS)

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.

This section applies to MRS 3.*x* or later.

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 17-144 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: **./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. 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.	-

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*

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.

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 17-145 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

17.16.9 Example for Using the Open-Source sqoop-shell Tool (Oracle-HBase)

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.

This section applies to MRS 3.*x* or later.

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 17-146 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: **./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. 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.	-

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*

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.

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 17-147 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

17.17 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 17-148 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**)
	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

Log Type	Log File Name	Description
	loader_ctl.log	Log file generated when Loader executes the service start and stop script (**sqoop.sh**)

Log Level

Table 17-149 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 17-149 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 17-150 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]}

17.18 Example: Using Loader to Import Data from OBS to HDFS

Scenario

If you need to import a large volume of data from the external cluster to the internal cluster, import it from OBS to HDFS.

Prerequisites

- You have prepared service data.
- You have created an analysis cluster.

Procedure

Step 1 Upload service data to your OBS file system.

Step 2 Obtain the AK/SK information and create an OBS and HDFS link.

For details, see **Loader Link Configuration**.

Step 3 Access the Loader page.

If Kerberos authentication is enabled in the analysis cluster, refer to instructions in **Accessing the Hue Web UI**.

Step 4 Click **New Job**.

Step 5 In **Information**, set parameters.

1. In **Name**, enter a job name. For example, **obs2hdfs**.
2. In **From link**, select the OBS link you create.
3. In **To link**, select the HDFS link you create.

Step 6 In **From**, set source link parameters.

1. In **Bucket Name**, enter a name of the OBS file system.
2. In **Input directory or file**, enter a detailed location of service data in the file system.

   If it is a single file, enter a complete path containing the file name. If it is a directory, enter the complete path of the directory.
3. In **File format**, enter the type of the service data file.

For details, see **obs-connector**.

Step 7 In **To**, set destination link parameters.

1. In **Output directory**, enter the directory for storing service data in HDFS.

   If Kerberos authentication is enabled in the cluster, the current user accessing Loader needs to have the permission to write data to the directory.
2. In **File format**, enter the type of the service data file.

   The type must correspond to the type in **Step 6.3**.
3. In **Compression codec**, enter a compression algorithm. For example, if you do not compress data, select **NONE**.
4. In **Overwrite**, select **True**.
5. Click **Show Senior Parameter** and set **Line Separator**.
6. Set **Field Separator**.

For details, see **hdfs-connector**.

Step 8 In **Task Config**, set job running parameters.

1. In **Extractors**, enter the number of Map tasks.
2. In **Loaders**, enter the number of Reduce tasks.

   If the destination link is an HDFS link, **Loaders** is hidden.
3. In **Max error records in single split**, enter an error record threshold.
4. In **Dirty data directory**, enter a directory for saving dirty data, for example, **/user/sqoop/obs2hdfs-dd**.

Step 9 Click **Save and execute**.

On the **Manage jobs** page, view the job running result. You can click **Refresh** to obtain the latest job status.

----End

17.19 Common Issues About Loader

17.19.1 Data Cannot Be Saved in Internet Explorer 10 or 11

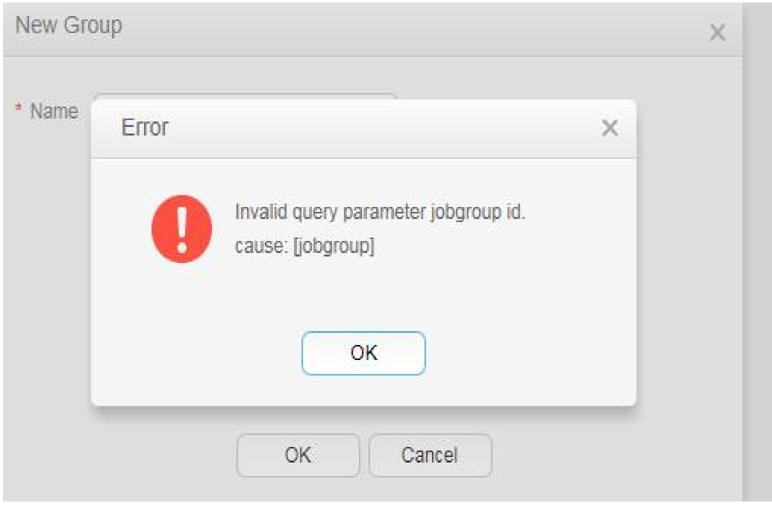
Question

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.

17.19.2 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.

18 Using Kudu

18.1 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

18.2 Accessing the Kudu Web UI

You can view Kudu job information on the Kudu web UI.

Prerequisites

Kudu has been installed in a cluster.

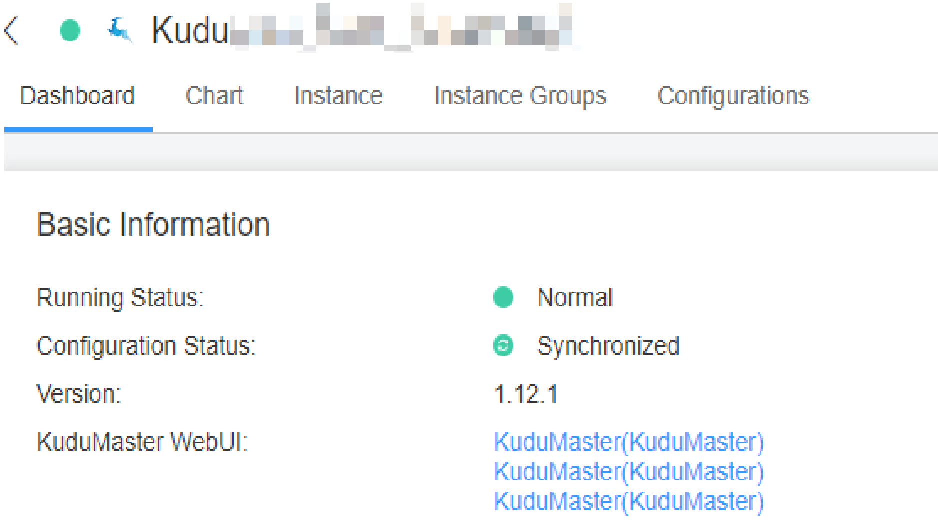
Accessing KuduMaster WebUI (MRS 3.x or Later)

Step 1 Log in to Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.

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 18-1 KuduMaster WebUI



----End

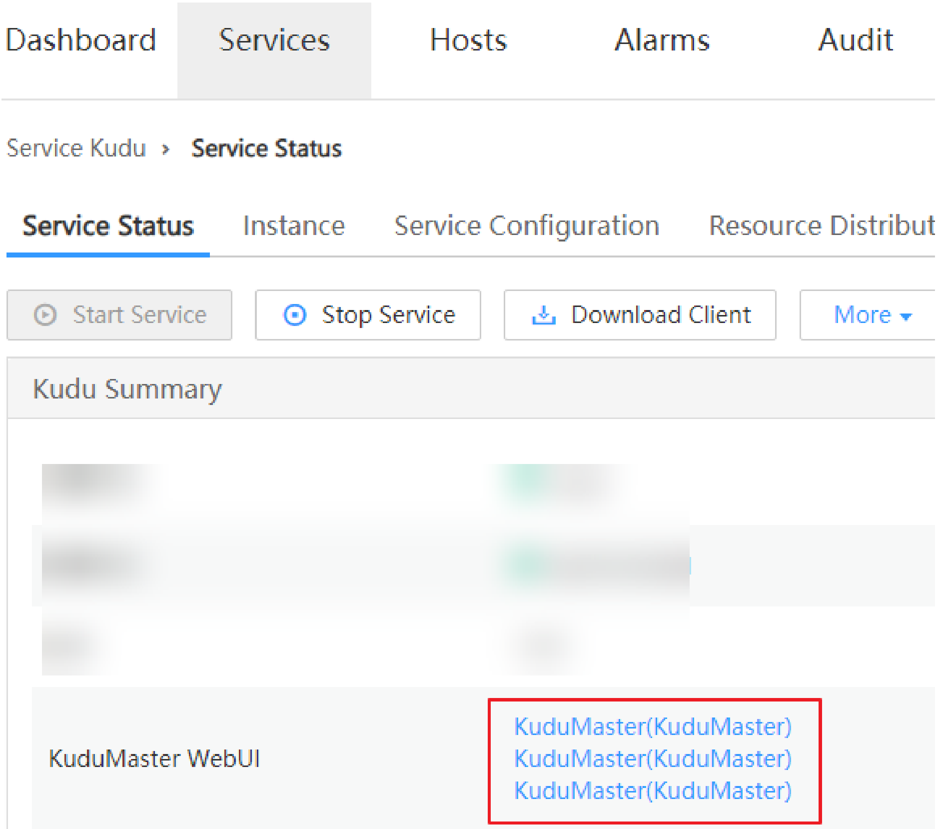
Accessing KuduMaster WebUI (Versions Earlier Than MRS 3.*x*)

Step 1 Access Manager. For details, see **Accessing MRS Manager (Versions Earlier Than MRS 3.x)**.

Step 2 choose **Services** > **Kudu**.

Step 3 In **KuduMaster WebUI** of **Kudu Summary**, click **KuduMaster(KuduMaster)**. The KuduMaster web UI is displayed.

Figure 18-2 KuduMaster WebUI



----End

19 Using MapReduce

19.1 Configuring the Distributed Cache to Execute MapReduce Jobs

Scenarios

NOTE

This section applies to MRS 3.*x* or later.

Distributed caching is useful in the following scenarios:

- **Rolling upgrade**

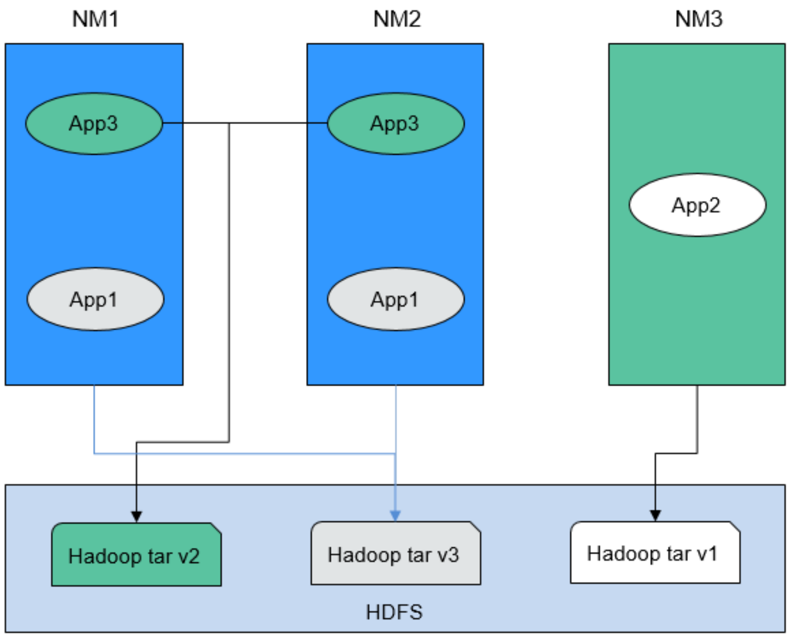
  During the upgrade, applications must keep the text content (JAR file or configuration file) unchanged. The content is not based on YARN of the current version, but based on the version when YARN is submitted. Generally, applications (such as MapReduce, Hive, and Tez) need to be installed locally. Libraries 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 19-1 Clusters with NodeManagers and Applications of multiple versions



As shown in **Figure 19-1**, 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 Log in to the HDFS client.

1. Log in to the node where the client is installed as the client installation user.
2. Go to the client installation directory.

   cd *Client installation path*
3. Configure environment variables.

   source bigdata_env
4. If the cluster is in security mode, authenticate the user. Skip user authentication for normal clusters.

   kinit *Component service user*

Step 2 Save the MapReduce **.tar** package of the specified version to a directory that can be accessed by applications in HDFS.

$HADOOP_HOME/**bin/hdfs dfs -put** *hadoop-x.tar.gz /mapred/framework/*

Step 3 Set parameters in the *Client installation path*/**Yarn/config/mapred-site.xml** file by referring to **Table 19-1**.

vi *Client installation path*/**Yarn/config/mapred-site.xml**

Table 19-1 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>	N/A
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 **.tar** packages 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 MapReduce tasks using the MapReduce **.tar** package of the *x* version:

hadoop jar *share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar pi* **-conf etc/hadoop-x/mapred-site.xml** *10 10*

----End

19.2 Configuring the MapReduce Shuffle Address

Scenario

When the MapReduce shuffle service is started, it attempts to bind an IP address based on local host. Configure the MapReduce shuffle service to connect 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 on the node where the NodeManager instance is deployed. Example path: **${BIGDATA_HOME}/FusionInsight_HD_***xxx/x_xx***_NodeManager/etc/mapred-site.xml**

Table 19-2 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.	-

19.3 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 19-3 Parameter description

Parameter	Description	Default Value
mapreduce.cluster.acls.enabled	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.	For versions earlier than MRS 3.x: mapred For MRS 3.*x* or later: mapred supergroup,System_administrator_186

19.4 Submitting a MapReduce Task on Windows

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:

NOTE

This section applies to MRS 3.*x* or later.

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 19-4 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

19.5 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 19-5 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 19-6 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. This parameter applies to MRS 3.*x*.	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**. This parameter applies to MRS 3.*x*.	-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.

19.6 MapReduce Performance Tuning

19.6.1 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 19-7 Settings of multiple CPU cores

Configuration	Parameter	Configuration Description
Number of slots in a node container	yarn.nodemanager.resource.memory-mb ● Description: total physical memory that can be used by YARN on a node. The unit is MB. ● Default value: Versions earlier than MRS 3.*x*: **8192** MRS 3.*x* or later: **16384** ● Parameter configuration entry: For versions earlier than MRS 3.*x*, configure this parameter on the MRS console. For MRS 3.*x* or later: You need to configure this parameter on FusionInsight Manager.	● The parameter combination determines the number of concurrent tasks (Map and Reduce tasks) on each node. ● When multiple processes access the same disk simultaneously due to all Map/Reduce tasks requiring reading and writing data to a disk, it causes 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 ● Description: memory limit of a Map task. The unit is MB. ● Default value: **4096** ● Parameter configuration entry: 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.	
	mapreduce.reduce.memory.mb ● Description: memory limit of a Reduce task. The unit is MB. ● Default value: **4096** ● Parameter configuration entry: 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.	

Configuration	Parameter	Configuration Description
Map output and compression	mapreduce.map.output.compress ● Description: output of a Map task can be compressed before being transmitted over the network. It is a per-job configuration. ● Default value: **true** ● Parameter configuration entry: 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.	● Compressing the Map task output before writing it to disks can provide benefits such as saving disk space, faster data write, and reduced data traffic delivered to the Reducer. You need to perform the configuration on the 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 ● Description: codec used for compression. ● Default value: **org.apache.hadoop.io.compress.Lz4Codec** ● Parameter configuration entry: 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.	
Spills	mapreduce.map.sort.spill.percent ● Description: soft limit in the serialization buffer. Once this limit is reached, the thread will begin to overflow content to disk in the background. ● Default value: **0.8** ● Parameter configuration entry: 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.	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	Parameter	Configuration Description
Data packet size	dfs.client-write-packet-size ● Description: size of the data packet. It can be specified by each job. ● Default value: **262144** ● Parameter configuration entry: 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.	● When the HDFS client writes data to a data node, the data will be accumulated until a packet is generated. The data packet is transmitted over the network. ● 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

19.6.2 Configuring the Baseline Parameters for MapReduce Jobs

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**.

  Table 19-8 Parameter configuration 1

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**.

Table 19-9 Parameter configuration 2

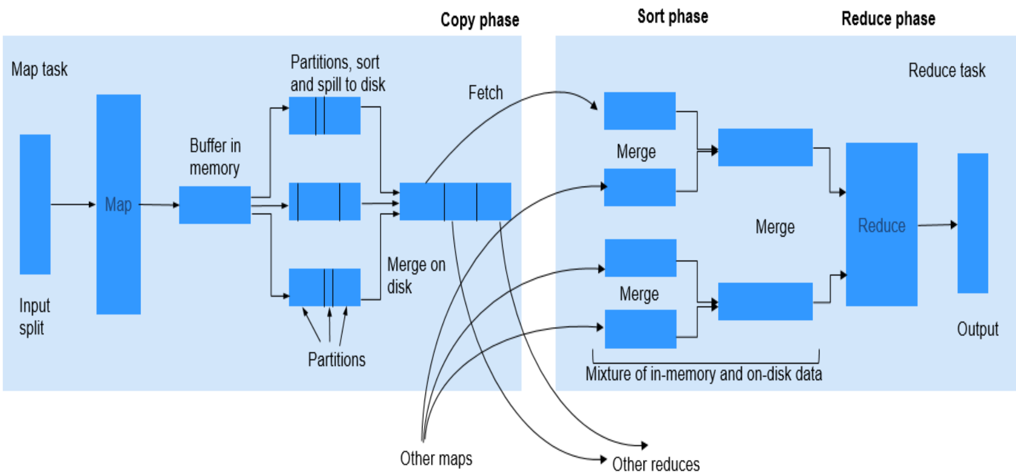
Parameter	Description	Default Value
mapreduce.input.fileinputformat.split.maxsize	Indicates the maximum size of the data block into which the Map input information is to be split. The shard size can be calculated based on its size customized by the user and the block size of each file. The formula is 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

19.6.3 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 19-2 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 19-10 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**.	For versions earlier than MRS 3.*x*: -Xmx2048M -Djava.net.preferIPv4Stack=true For MRS cluster 3.*x* and later versions: • 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 19-11 Combiner API

Class	API	Description
org.apache.hadoop.mapreduce.Job	public void setCombinerClass(Class<? extends Reducer> cls)	This is 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 19-12 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

19.6.4 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.	• For versions earlier than MRS 3.x: -Xmx1024m -XX:CMSFullGCsBeforeCompaction=1 -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:+UseCMSCompactAtFullCollection -verbose:gc • MRS 3.x or later: -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

19.6.5 Configuring Speculative Execution for MapReduce Tasks

Scenario

Clusters with hundreds or thousands of nodes are more prone to software and hardware issues, resulting in longer task execution times. This can cause tasks to be delayed until a malfunctioning device is fixed. In speculative execution, a task is allocated to multiple machines for running, and the node that completes the task first is selected. 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

19.6.6 Tuning MapReduce Tasks 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

19.6.7 Optimizing the Commit Phase of MapReduce Tasks

Scenario

By default, if a MapReduce job generates a large number of output files, it takes a long time for the job to commit task temporary results to the final output directory in the last 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 19-13 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

19.6.8 Improving MapReduce Client Task Reliability

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 19-14 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. This parameter applies to MRS 3.*x*.	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

19.7 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 19-15 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
	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

Type	Name	Description
	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 19-16 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 19-16 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 19-17 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)
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)

19.8 Common Issues About MapReduce

19.8.1 How Do I Handle the Problem that MapReduce Task Has No Progress for a Long Time?

Question

The MapReduce task has no progress for a long time.

Answer

Generally, this is caused by insufficient memory. If the memory is small, it takes a long time to copy the map output.

To reduce the waiting time, increase the heap memory.

You can optimize task configuration 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**

For example, if the data size of 10 mappers is 5 GB, the ideal heap memory is 1.5 GB. Increase the heap memory as the data size increases.

19.8.2 Why Is the Client Unavailable When a Task Is 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**.

19.8.3 What Should I Do If HDFS_DELEGATION_TOKEN Cannot Be Found in the Cache?

Question

Why cannot **HDFS_DELEGATION_TOKEN** be found in the cache in a cluster with Kerberos authentication enabled?

Answer

In MapReduce, **HDFS_DELEGATION_TOKEN** is deleted by default after a task is complete. If **HDFS_DELEGATION_TOKEN** is used in the next task, it will not be found in the cache.

To use the same token again in subsequent work, configure parameters for MapReduce tasks. If this parameter is set to **false**, the user can use the same token again.

```
jobConf.setBoolean("mapreduce.job.complete.cancel.delegation.tokens", false);
```

19.8.4 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/

19.8.5 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. Optimize configuration of 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**.

19.8.6 What Should I Do If MapReduce Job Information Cannot Be Opened Through Tracking URL on the ResourceManager Web UI?

Question

Why cannot a finished MapReduce job be opened on the ResourceManager web UI after the MapReduce JobHistoryServer service address is changed?

Answer

The JobHistoryServer address (**mapreduce.jobhistory.address / mapreduce.jobhistory.webapp.<https.>address**) is a MapReduce parameter that is submitted to the ResourceManager along with the job. Once the job is finished, the ResourceManager saves this parameter as the redirection address for viewing job history information in RMStateStore.

Update the MapReduce client configuration file with the new JobHistoryServer service address promptly to view the job history information of a new job. Otherwise, the job history information page will not load as it will still point to the old address. For MapReduce jobs that ran before the service address was changed, the redirection information is stored in RMStateStore and cannot be modified. Therefore, the ResourceManager web UI cannot redirect to the new address. However, you can still search for jobs by directly accessing the new JobHistoryServer service address, and the job information will remain intact.

19.8.7 Why MapReduce Tasks Fails in the Environment with Multiple NameServices?

Question

Why MapReduce or YARN tasks using the viewFS function fail to be executed in the environment with multiple NameServices?

Answer

When viewFS is used, only directories mounted to viewFS can be accessed. Therefore, the most possible reason is that the configured path is not on the mount point of viewFS. The following is an 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>
```

In the MapReduce configuration that depends on HDFS, the mounted directory needs to be used.

Incorrect example:

```
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>/tmp/hadoop-yarn/staging</value>
</property>
```

The root directory (/) cannot be accessed in viewFS.

Correct example:

```
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
```

```
<value>/folder1/tmp/hadoop-yarn/staging</value>
</property>
```

19.8.8 What Should I Do If the Partition-based Task Blacklist Is Abnormal?

Question

The Map&Reduce task fails, and the ratio of the number of faulty nodes to the total number of nodes in the cluster is lower than the blacklist threshold specified by **yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold**. Why is the faulty node not added to the blocklist?

Answer

If the number of blocked nodes exceeds the threshold, all blocked nodes are released. The threshold is based on the ratio of fault nodes to all nodes in the cluster. Currently, each node has a label expression. The blocklist threshold is calculated based on the number of nodes relate to effective node labels. In other way, the blocklist threshold is the ratio of fault nodes to relate to effective node labels.

Assume that there are 100 nodes in the cluster, including 10 nodes (labelA) with valid node label expressions. Assume that all nodes relate to valid node label expressions are faulty and default blocklist threshold is 0.33. After calculation, 10/100 = 0.1, which is far smaller than the threshold. In this case, the 10 nodes will never get released. Therefore, MapReduce tasks always cannot obtain nodes and applications cannot run properly. In practice, the threshold needs to be calculated based on the total number of nodes relate to valid node label expressions: 10/10 = 1 is greater than the blacklist release threshold and all nodes are released.

Therefore, even the ratio of fault nodes to all nodes in the clusters is within the threshold, all nodes in the blocklist are released.

20 Using OpenTSDB

20.1 Using an MRS Client to Operate OpenTSDB Metric Data

You can perform an interactive operation on an MRS cluster client. For a cluster with Kerberos authentication enabled, the user must belong to the **opentsdb**, **hbase**, **opentsdbgroup**, and **supergroup** groups and have the HBase permission.

Prerequisites

- The password of user **admin** has been obtained. The password of user **admin** is specified by the user during MRS cluster creation.
- The cluster client has been installed, for example, in the **/opt/client** directory. The client directory in the following operations is only an example. Change it based on the actual installation directory onsite. For details about how to update a client, see **Updating a Client (Versions Earlier Than 3.x)**.

Using a Client

Step 1 If Kerberos authentication is enabled for the current cluster, log in to MRS Manager and create a user that belongs to the **opentsdb**, **hbase**, **opentsdbgroup**, and **supergroup** groups and has the HBase permission, for example, **opentsdbuser**. If Kerberos authentication is disabled for the current cluster, skip this step.

Step 2 Prepare a client based on service conditions and log in to the node where the client is installed.

For example, if you have updated the client on the Master2 node, log in to the Master2 node to use the client. For details, see **Updating a Client (Versions Earlier Than 3.x)**.

Step 3 Run the following command to switch the user:

sudo su - omm

Step 4 Run the following command to switch to the client directory, for example, **/opt/client**.

cd /opt/client

Step 5 Run the following command to configure environment variables:

source bigdata_env

Step 6 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.

- If the user is a human-machine user, run the **kinit opentsdbuser** command to authenticate the user.
- If the user is a machine-machine user, download the user authentication credential file, and save and decompress it to obtain the user's **user.keytab** and **krb5.conf** files. Go to the decompressed **user.keytab** directory, and run the **kinit -kt user.keytab opentsdbuser** command to authenticate the user.

Step 7 Operate the OpenTSDB data. For details, see **Operating Data**.

----End

Operating Data

- Viewing help information

  Run the **tsdb** command to print all commands supported by OpenTSDB, for example, **fsck**, **import**, **mkmetric**, **query**, **tsd**, **scan**, **search**, **uid**, and **version**.

  Command output:

```
tsdb: error: unknown command ''
usage: tsdb <command> [args]
Valid commands: fsck, import, mkmetric, query, tsd, scan, search, uid, version
```

- Creating an OpenTSDB metric

  Run the **tsdb mkmetric** command to create a metric. For example, run the **tsdb mkmetric sys.cpu.user** command to create a metric named **sys.cpu.user**.

  Command output:

```
Start run net.opentsdb.tools.UidManager, args: assign metrics sys.cpu.user
metrics sys.cpu.user: [0, 0, 6]
```

- Importing data to the OpenTSDB metric
  a. Prepare a metric file, for example, the **importData.txt** file that contains following information.

     sys.cpu.user 1356998400 41 host=web01 cpu=0

     sys.cpu.user 1356998401 42 host=web01 cpu=0

     sys.cpu.user 1356998402 44 host=web01 cpu=0

     sys.cpu.user 1356998403 47 host=web01 cpu=0

     sys.cpu.user 1356998404 42 host=web01 cpu=0

     sys.cpu.user 1356998405 42 host=web01 cpu=0
  b. Run the **tsdb import** command to import metric data. For example, run the **tsdb import importData.txt** command to import the **importData.txt** file.

```
Start run net.opentsdb.tools.TextImporter, args: importData.txt
2019-06-26
15:45:22,091 INFO  [main] TextImporter:
```

```
reading from file:importData.txt
2019-06-26
15:45:22,102 INFO  [main] TextImporter:
Processed importData.txt in 11 ms, 6 data points (545.5 points/s)
2019-06-26
15:45:22,102 INFO  [main] TextImporter:
Total: imported 6 data points in 0.012s (504.0 points/s)
```

- Querying the OpenTSDB metric

  Run the **tsdb uid metrics** command to obtain the metric stored in OpenTSDB. For example, run the **tsdb uid metrics sys.cpu.user** command to query the data of the **sys.cpu.user** metric.

  Command output:

```
Start run net.opentsdb.tools.UidManager, args: metrics sys.cpu.user
metrics sys.cpu.user: [0, 0, 6]
```

  To obtain more information, run the **tsdb uid** command.

```
Start run net.opentsdb.tools.UidManager, args:
Not enough arguments
Usage: uid <subcommand> args
Sub commands:
  grep [kind] <RE>: Finds matching IDs.
  assign <kind> <name> [names]: Assign an ID for the given name(s).
  rename <kind> <name> <newname>: Renames this UID.
  delete <kind> <name>: Deletes this UID.
  fsck: [fix] [delete_unknown] Checks the consistency of UIDs.
        fix            - Fix errors. By default errors are logged.
        delete_unknown - Remove columns with unknown qualifiers.
                         The "fix" flag must be supplied as well.
  [kind] <name>: Lookup the ID of this name.
  [kind] <ID>: Lookup the name of this ID.
  metasync: Generates missing TSUID and UID meta entries, updates created timestamps
  metapurge: Removes meta data entries from the UID table
  treesync: Process all timeseries meta objects through tree rules
  treepurge <id> [definition]: Purge a tree and/or the branches from storage. Provide an integer Tree
ID and                                                    optionally add "true" to delete the tree definition
Example values for [kind]: metrics, tagk (tag name), tagv (tag value).
  --config=PATH    Path to a configuration file (default: Searches for file see docs).
  --idwidth=N      Number of bytes on which the UniqueId is encoded.
  --ignore-case    Ignore case distinctions when matching a regexp.
  --table=TABLE    Name of the HBase table where to store the time series (default: tsdb).
  --uidtable=TABLE Name of the HBase table to use for Unique IDs (default: tsdb-uid).
  --verbose        Print more logging messages and not just errors.
  --zkbasedir=PATH Path under which is the znode for the -ROOT- region (default: /hbase).
  --zkquorum=SPEC  Specification of the ZooKeeper quorum to use (default: localhost).
  -i               Short for --ignore-case.
  -v               Short for --verbose.
```

- Scanning the OpenTSDB metric data

  Run the **tsdb query** command to query the imported metric data in batches. The command format is as follows: ***tsdb query <START-DATE> <END-DATE> <aggregator> <metric> <tagk=tagv>***. For example, run the **tsdb query 0 1h-ago sum sys.cpu.user host=web01** command.

```
Start run net.opentsdb.tools.CliQuery, args: 0 1h-ago sum sys.cpu.user host=web01
sys.cpu.user 1356998400000 41 {host=web01, cpu=0}
sys.cpu.user 1356998401000 42 {host=web01, cpu=0}
sys.cpu.user 1356998402000 44 {host=web01, cpu=0}
sys.cpu.user 1356998403000 47 {host=web01, cpu=0}
sys.cpu.user 1356998404000 42 {host=web01, cpu=0}
sys.cpu.user 1356998405000 42 {host=web01, cpu=0}
```

NOTE

<START-DATE>: start time of the metric to be queried

<END-DATE>: end time of the metric to be queried

<aggregator>: aggregation mode of the data query

<metric>: name of the metric to be queried

<tagk=tagv>: key and value of a tag

- Deleting the imported OpenTSDB metric

  Run the **tsdb uid delete** command to delete the imported metric and its value. For example, to delete the **sys.cpu.user** metric, run the **tsdb uid delete metrics sys.cpu.user** command.

  ```
  Start run net.opentsdb.tools.UidManager, args: delete metrics sys.cpu.user
  ```

20.2 Running the curl Command to Operate OpenTSDB

Writing Data

For example, to write data of a metric named **testdata**, whose timestamp is **1524900185**, value is **true**, tag is **key** and **value**, run the following command:

```
curl -ki -X POST -d '{"metric":"testdata", "timestamp":1524900185, "value":"true", "tags":
{"key":"value"}}' https://<tsd_ip>:4242/api/put?sync
```

<tsd_ip>: indicates the IP address of the TSD instance of OpenTSDB to which data is to be written.

```
HTTP/1.1 204 No Content
Content-Type: application/json; charset=UTF-8
Content-Length:0
```

Querying Data

For example, to query summary information about the **testdata** metric in the past three years, run the following command:

```
curl -ks https://<tsd_ip>:4242/api/query?start=3y-ago\&m=sum:testdata | python -m json.tool
```

- **<tsd_ip>**: indicates the IP address or host name of the TSD instance of OpenTSDB that needs to be accessed.
- **<start=3y-ago\&m=sum:testdata>**: Translates the **&** symbol, which may not be identified in the request.
- (Optional) **<python -m json.tool>**: Converts the response request to the JSON format.

```
[
    {
        "aggregateTags": [],
        "dps": {
            "1524900185": 1
        },
        "metric": "testdata",
        "tags": {
            "key": "value"
        }
    }
]
```

Querying tsd Status

For example, to query information about the client connected to HBase, run the following command:

```
curl -ks https://<tsd_ip>:4242/api/stats/region_clients | python -m json.tool
```

<tsd_ip>: indicates the IP address of the TSD instance of OpenTSDB that needs to be accessed.


```
[
    {
        "dead": false,
        "endpoint":"/xx.xx.xx.xx:16020",
        "inflightBreached": 0,
        "pendingBatchedRPCs": 0,
        "pendingBreached": 0,
        "pendingRPCs": 0,
        "rpcResponsesTimedout": 0,
        "rpcResponsesUnknown": 0,
        "rpcid": 78,
        "rpcsInFlight": 0,
        "rpcsSent": 79,
        "rpcsTimedout": 0,
        "writesBlocked": 0
    }
]
```

21 Using Oozie

21.1 Using Oozie Client to Submit an Oozie Job

21.1.1 Oozie Client Configurations

Scenario

This section describes how to use the Oozie client in an O&M scenario or service scenario. Oozie can be used to submit a wide array of jobs, such as Hive, Spark2x, Loader, MapReduce, Java, DistCp, Shell, HDFS, SSH, SubWorkflow, Streaming, and scheduled jobs.

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. Replace it with the actual installation directory.
- 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.

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 Submit a MapReduce job on the Oozie client.

1. Modify the task execution configuration file.
   cd /opt/client/Oozie/oozie-client-*/examples/apps/map-reduce/
   vi job.properties

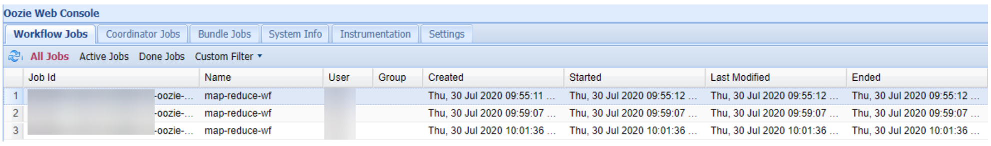
```
nameNode=hdfs://hacluster
resourceManager=10.64.35.161:8032 (10.64.35.161 is the service plane IP address of the YARN resourceManager (active) node, and 8032 is the port number of yarn.resourcemanager.port)
queueName=default
examplesRoot=examples
user.name=admin
oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/apps/map-reduce# HDFS upload path
outputDir=map-reduce
oozie.wf.rerun.failnodes=true
```

2. Execute the Oozie job.
   oozie job -oozie https://*Host name of the Oozie role***:21003/oozie/ -config job.properties -run**
   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**.

```
[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, click **Cluster**, click the name of the desired cluster, and choose **Services** > **Oozie**. Click the hyperlink next to **Oozie WebUI** to go to the Oozie web UI and view the task execution result.

Figure 21-1 Task execution result



----End

21.1.2 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 21-1 lists the files that you need to pay attention to in the directory.

Table 21-1 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

21.1.3 Submitting a Spark2xTask 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 21-2 lists the files that you need to pay attention to in the directory.

Table 21-2 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

21.1.4 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 21-3 lists the files that you need to pay attention to in the directory.

Table 21-3 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

21.1.5 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 21-4 lists the files that you need to pay attention to in the directory.

Table 21-4 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

21.1.6 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 21-5 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 21-6 lists the files that you need to pay attention to in the example directory.

Table 21-6 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

21.2 Using Hue to Submit an Oozie Job

21.2.1 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.

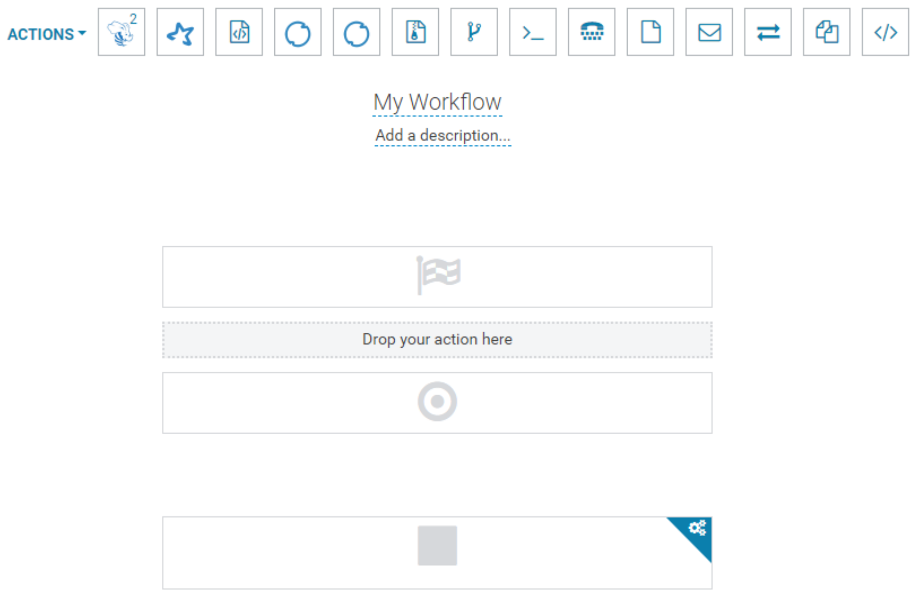
For example, log in to FusionInsight Manager as user **admin** and choose **System** in the top menu bar. On the **System** page that is displayed, choose **User** under **Permission** in the navigation pane on the left. On the displayed **User** page, click

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

21.2.2 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

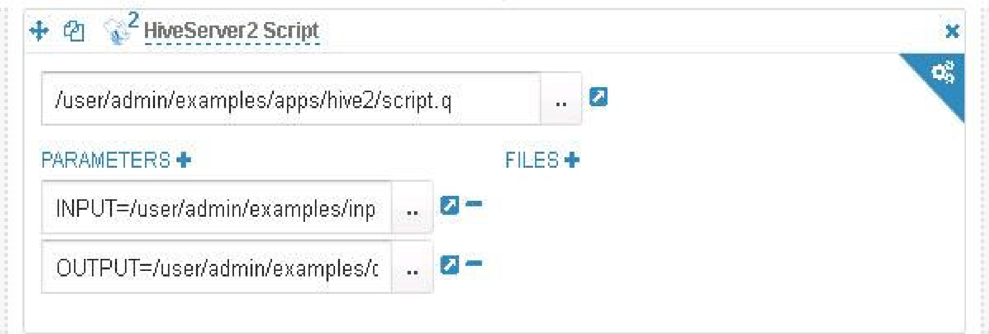
Step 1 Create a workflow. For details, see **Creating a Workflow Using Hue**.

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**.

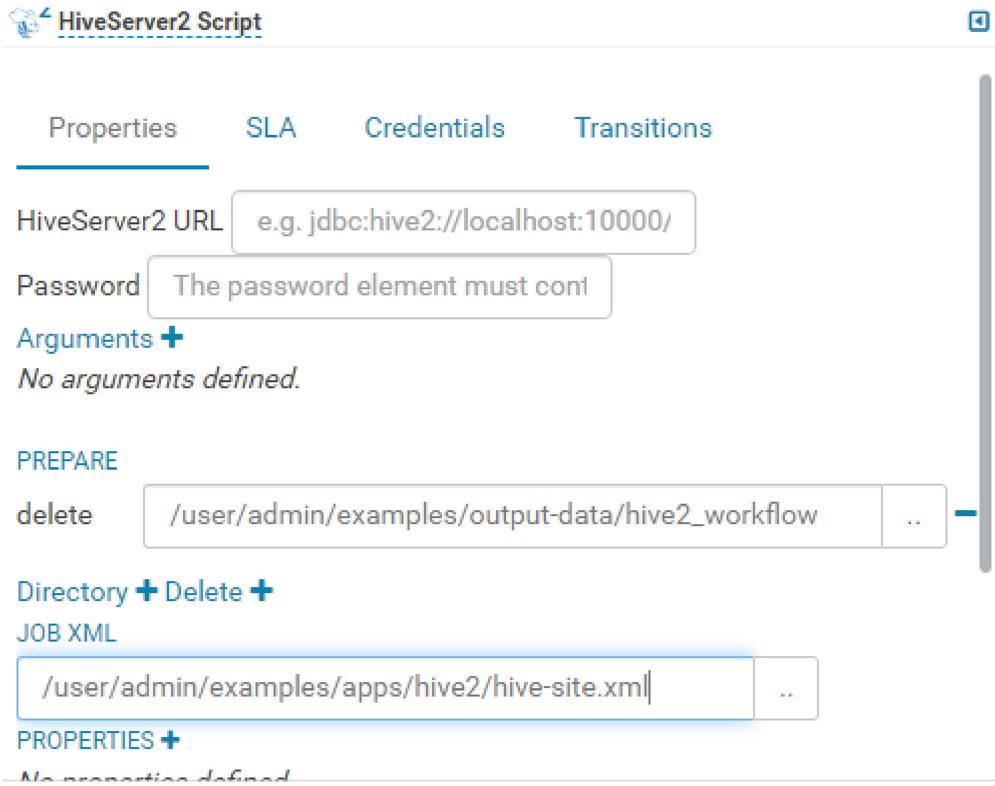
Step 4 Click **PARAMETER+** to add input and output parameters.

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 the **JOB XML**. The value is *Client installation directory*/**Oozie/oozie-client-*/examples/apps/hive/hive-site.xml**. For example, to upload the file to the HDFS directory, set the value to **/user/admin/examples/apps/hive2/hive-site.xml**. You do not need to set **HiveServer2 URL** or 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

21.2.3 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 HiveQL Editor on the Hue Web UI**. 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

21.2.4 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/8.1.0.1/spark-archive-2x.zip --conf spark.eventLog.enabled=true --conf spark.eventLog.dir=hdfs://hacluster/spark2xJobHistory2x**.

NOTE

The version 8.1.0.1 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

21.2.5 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

21.2.6 Submitting an Oozie Loader Job Using Hue

Scenario

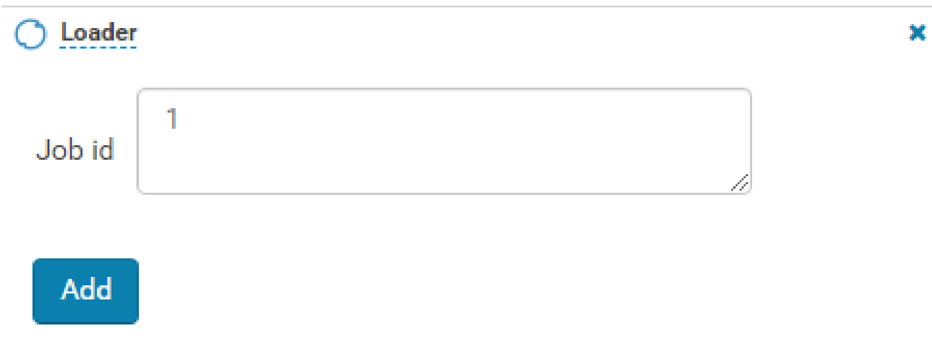
This section describes how to submit an Oozie job of the Loader type on the Hue web UI.

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

21.2.7 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.

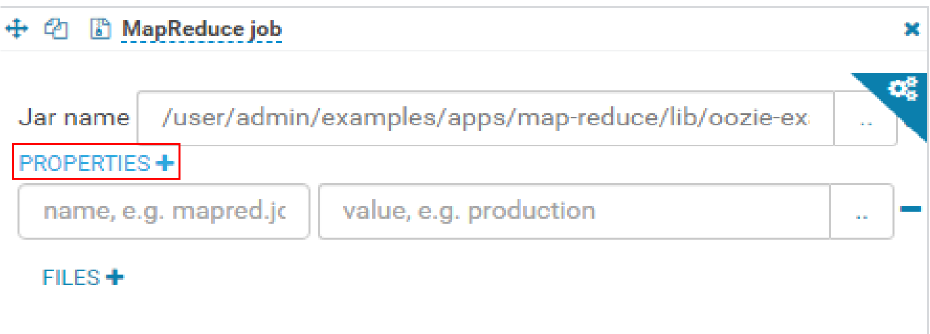
Procedure

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

21.2.8 Submitting an Oozie Sub Workflow Job Using Hue

Scenario

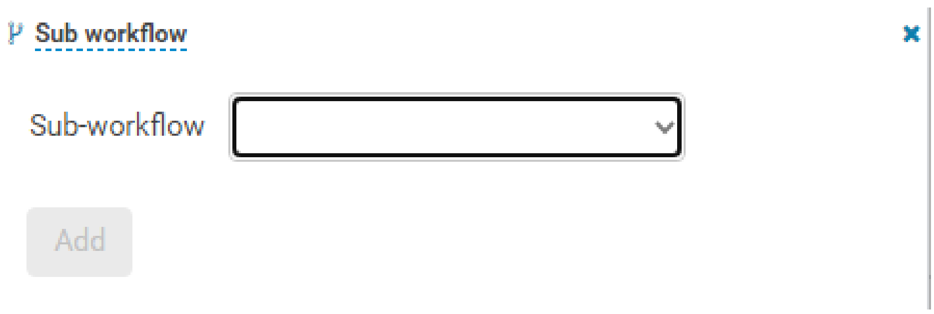
This section describes how to submit an Oozie job of the Sub-workflow type on the Hue web UI.

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

21.2.9 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

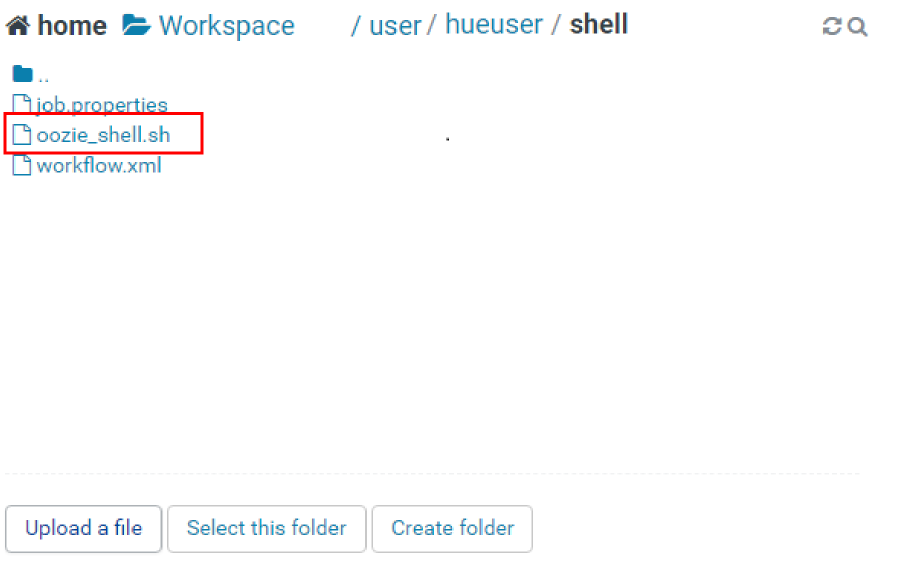
Step 1 Create a workflow. For details, see **Creating a Workflow Using Hue**.

Step 2 On the workflow editing page, select next to **Shell** and drag it to the operation area.

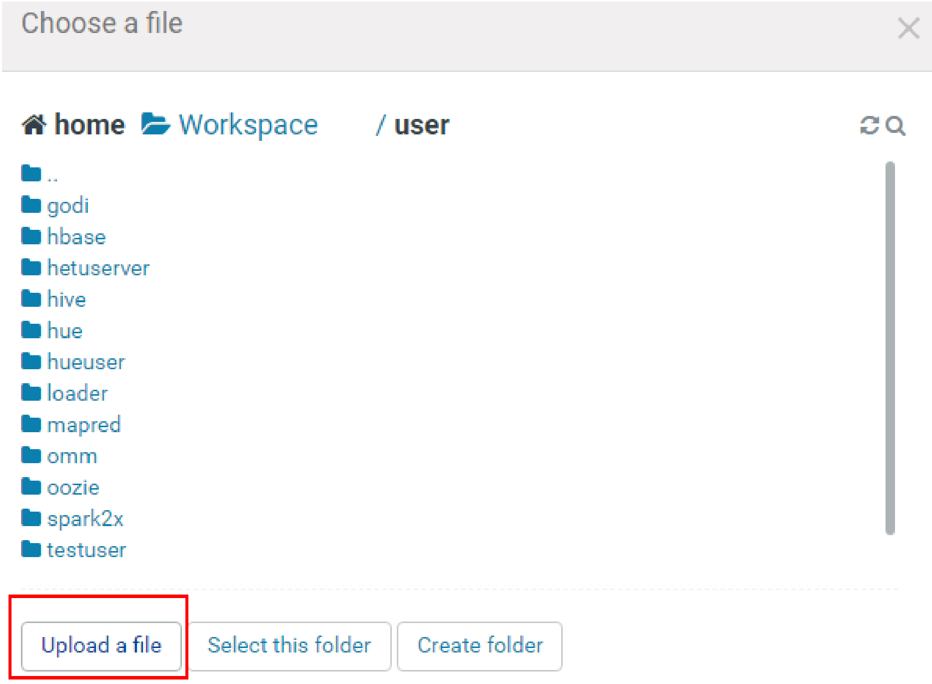
Step 3 In the **Shell** window that is displayed, set **Shell command**, for example, to **oozie_shell.sh**, and click **Add**.

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.

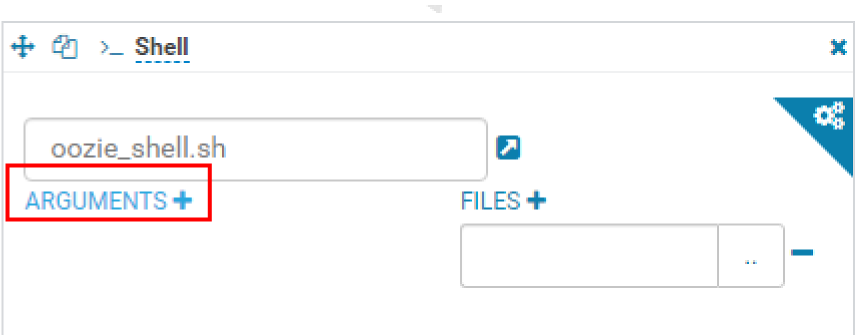
- If the file is stored in HDFS, select the path of the **.sh** file, for example, **user/hueuser/shell/oozie_shell.sh**.



- 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

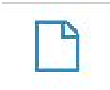
21.2.10 Submitting an Oozie HDFS Job Using Hue

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 delete HDFS directories, this parameter cannot be left empty. Otherwise, the **/user/{***User who submit the job***}** directory of the HDFS will be deleted by default, which may cause running errors of other tasks.

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

21.2.11 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

21.2.12 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

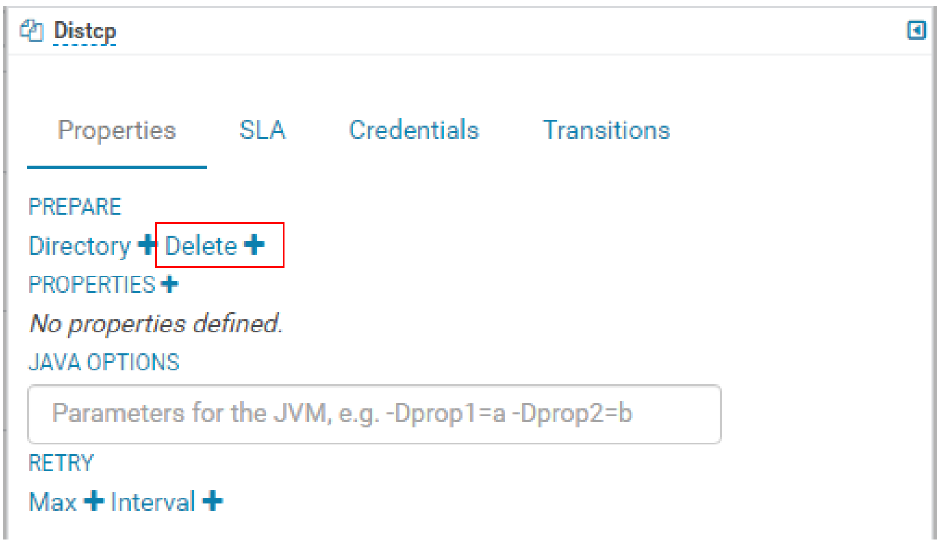
source_port: port number of the HDFS NameNode in the source cluster.

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

21.2.13 Submitting an Oozie SSH Job Using Hue

Scenario

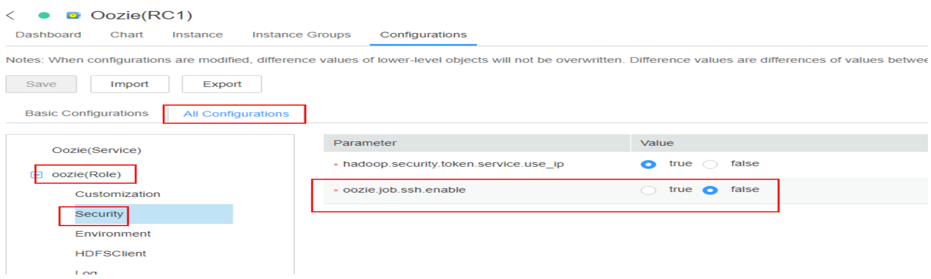
This section guides you to submit an Oozie job of the SSH type on the Hue web UI.

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 displayed **Ssh** window, set the following parameters and click **Add**.

- User in the User and Host configuration item is the user for whom mutual trust is configured in **Step 3**. The format is *User for running the SSH task@IP address of the node where the SSH task is running*. For example, the value of this configuration item can be **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

21.2.14 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.

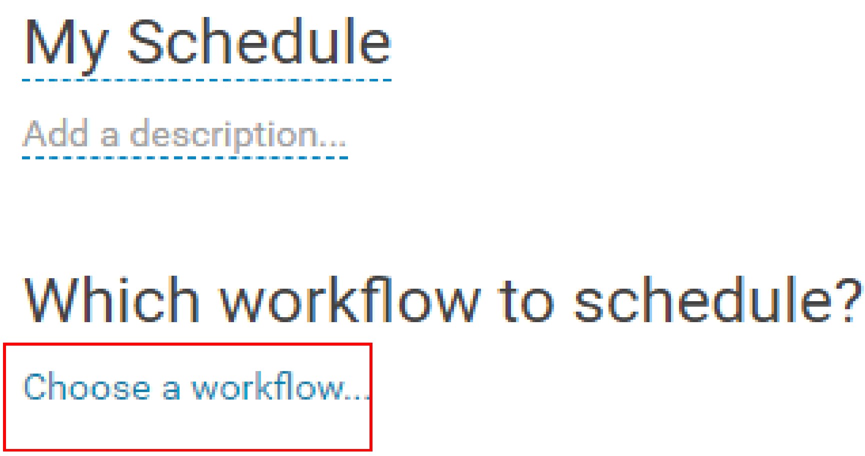
Procedure

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.

----End

21.2.15 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**.

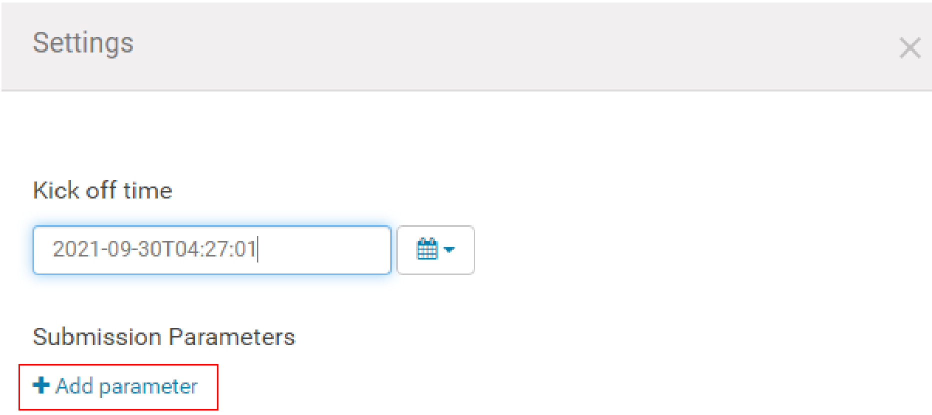
Step 2 In the navigation tree on the left, click and choose **Bundle** to open the Bundle editor.

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.

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

21.2.16 Querying Oozie Job Results on the Hue Web UI

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 21-7 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

21.2.17 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

21.3 Enabling 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

This section applies only to MRS 3.1.2 or later.

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

21.4 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 21-8 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 21-9 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 21-9 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 21-10 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$StatusTransitRunnable: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)

21.5 Common Issues About Oozie

21.5.1 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

Use UTC time. For example, set **start=2016-12-20T09:00Z** in **job.properties** file.

21.5.2 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

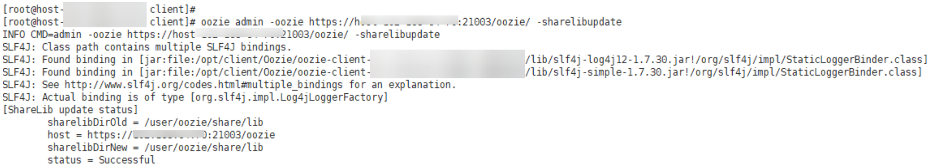
Run the following command on the client to refresh the directory:

oozie admin -oozie https://xxx.xxx.xxx.xxx:21003/oozie -sharelibupdate

21.5.3 Common Troubleshooting Methods for Oozie Job Execution Failures

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.



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.

22 Using Presto

22.1 Accessing the Presto Web UI

You can view the Presto statistics on the graphical Presto web UI. You are advised to use Google Chrome to access the Presto web UI because it cannot be accessed using Internet Explorer.

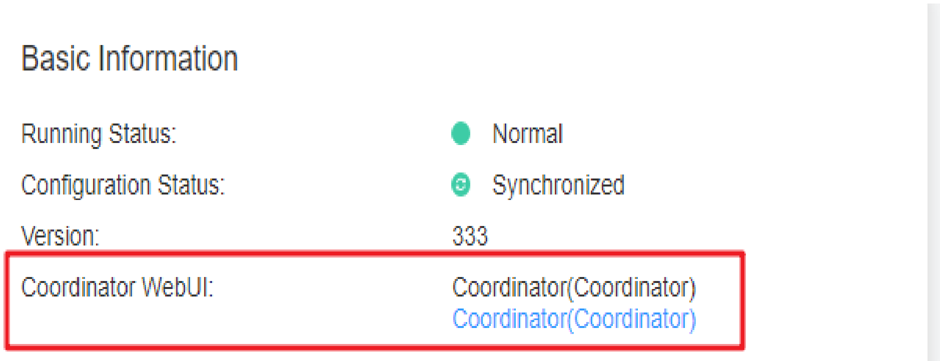
Prerequisites

- Presto has been installed in a cluster.
- The cluster 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 the actual installation directory onsite.

Accessing the Presto Web UI

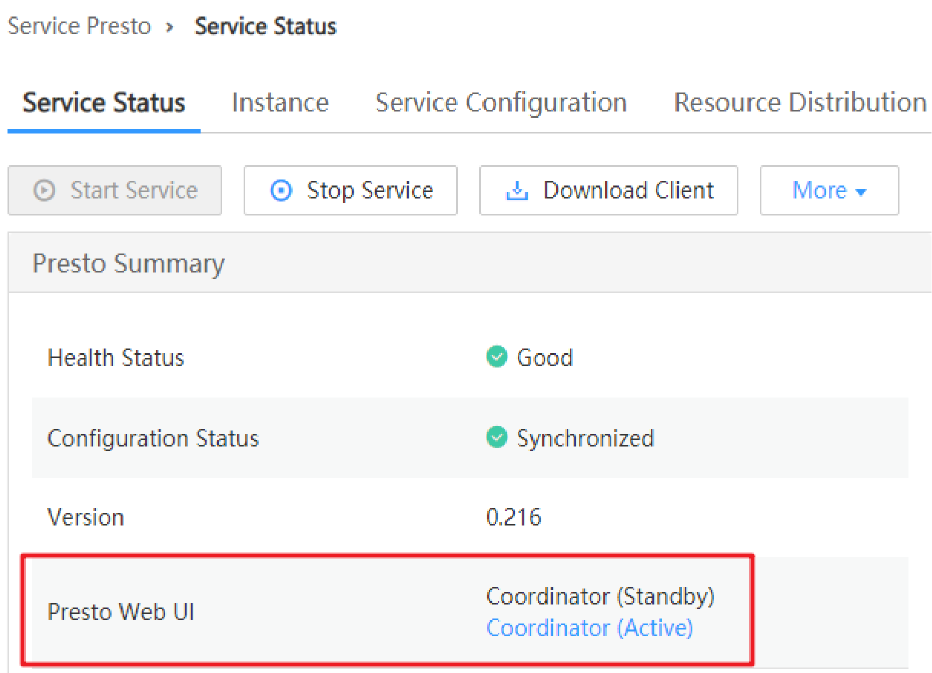
- Method 1 (for MRS 3.*x* or later)
  a. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > *Name of the target cluster* > **Services**.
  b. Select **Presto**. In the **Basic Information** area, click **Coordinator(Coordinator)** next to **Coordinator WebUI**. The Coordinator web UI is displayed.

Figure 22-1 Coordinator WebUI



- Method 2 (for versions earlier than MRS 3.*x*)
  a. Log in to MRS Manager and choose **Services**.
  b. Select **Presto**. In the **Presto Summary** area, click **Coordinator (Active)** next to **Presto Web UI**. The Presto web UI is displayed.

Figure 22-2 Presto WebUI



NOTE

When accessing the Presto web UI for the first time, you must add the address to the trusted site list.

- Method 3 (for MRS 1.9.2 or later)
  a. Log in to the MRS console, click the target cluster name to go to the cluster details page, and click the **Components** tab.

  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.)

  b. Click **Presto**. In the **Presto Summary** area, click **Coordinator (Active)** next to **Presto Web UI**. The Presto web UI is displayed.

22.2 Using a Client to Execute Query Statements

You can perform an interactive query on an MRS cluster client. For clusters with Kerberos authentication enabled, users who submit topologies must belong to the **presto** group.

The Presto component of MRS 3.x does not support Kerberos authentication.

Prerequisites

- The password of user **admin** has been obtained. The password of user **admin** is specified by the user during MRS cluster creation.
- The client has been updated.
- The Presto client has been manually installed for MRS 3.*x* clusters.

Procedure

Step 1 For clusters with Kerberos authentication enabled, log in to MRS Manager and create a role with the **Hive Admin Privilege** permission. For details about how to create a role, see **Creating a Role**

Step 2 Create a user to the **Presto** and **Hive** groups, bind the role created in **Step 1** to the user, and download the user authentication file. For details, see **Creating a User** and **Downloading a User Authentication File**.

Step 3 Upload the downloaded **user.keytab** and **krb5.conf** files to the node where the MRS client resides.

NOTE

For clusters with Kerberos authentication enabled, **Step 2** to **Step 3** must be performed. For normal clusters, start from **Step 4**.

Step 4 Prepare a client based on service conditions and log in to the node where the client is installed.

For example, if you have updated the client on the Master2 node, log in to the Master2 node to use the client. For details, see **Using an MRS Client**.

Step 5 Run the following command to switch the user:

sudo su - omm

Step 6 Run the following command to switch to the client directory, for example, **/opt/client**.

cd /opt/client

Step 7 Run the following command to configure environment variables:

source bigdata_env

Step 8 Connect to the Presto Server. The following provides two client connection methods based on the client type.

- Using the client provided by MRS
  - For clusters with Kerberos authentication disabled, run the following command to connect to the Presto Server of the cluster:

    presto_cli.sh
  - For clusters with Kerberos authentication disabled, run the following command to connect to the Presto Server of other clusters. In the command, **ip** indicates the floating IP address of the cluster Presto Server, which can be obtained by searching for **PRESTO_COORDINATOR_FLOAT_IP** in the Presto configuration items. **port** indicates the Presto Server port number and is set to **7520** by default.

presto_cli.sh --server *http://ip:port*

– For clusters with Kerberos authentication enabled, run the following command to connect to the Presto Server of the cluster:

presto_cli.sh --krb5-config-path *krb5.conf file path* --krb5-principal *User's principal* --krb5-keytab-path *user.keytab file path* --user *presto username*

– For clusters with Kerberos authentication enabled, run the following command to connect to the Presto Server of other clusters. In the command, **ip** indicates the floating IP address of the cluster Presto Server, which can be obtained by searching for **PRESTO_COORDINATOR_FLOAT_IP** in the Presto configuration items. **port** indicates the Presto Server port number and is set to **7521** by default.

presto_cli.sh --krb5-config-path *krb5.conf file path* --krb5-principal *User's principal* --krb5-keytab-path *user.keytab file path* --server *https://ip:port* --krb5-remote-service-name *Presto Server name*

- Using the native client

  The native client of Presto is **Presto/presto/bin/presto** in the client directory.

Step 9 Run a query statement, for example, **show catalogs;**.

NOTE

For clusters with Kerberos authentication enabled, when querying **Hive Catalog** data, the user who runs the Presto client must have the permission to access Hive tables and run the **grant all on table [table_name] to group hive;** command in Hive beeline to grant permissions to the Hive group.

Step 10 After the query is complete, run the following command to exit the client:

quit;

----End

22.3 Presto FAQ

22.3.1 How Do I Configure Multiple Hive Connections for Presto?

Question

How do I configure multiple Hive connections for Presto?

Procedure

Step 1 Copy and distribute the **core-site.xml** and **hdfs-site.xml** files of the target Hive cluster to the Presto cluster and place them in a path (for example, **/home/omm**) on which user **omm** has the read permission. Change the file owner to **omm:wheel** and file permission to **750**.

Step 2 Go to the Presto configuration page.

- For MRS 1.8.10 or earlier: Log in to MRS Manager by referring to **Accessing MRS Manager**. Choose **Services** > **Presto** > **Service Configuration**, and select **All** from the **Basic** drop-down list.
- For versions later than MRS 1.8.10 and version MRS 2.*x*: Click the cluster name on the MRS console, choose **Components** > **Presto** > **Service Configuration**, and select **All** from the **Basic** drop-down list.

  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.)
- For MRS 3.*x* or later: Log in to FusionInsight Manager and choose **Cluster** > *Name of the desired cluster* > **Services** > **Presto**. Click **Configurations** then **All Configurations**.

Step 3 Search for **connector-customize** in the search box.

Step 4 Add a connector named **myhive**.

Add the following configuration to **connector-customize**:

myhive.connector.name=hive-hadoop2

myhive.hive.metastore.uri=thrift://{*Metastore server IP address of the target Hive cluster*}:{*Metastore server port of the target Hive cluster*} (Use commas (,) to separate multiple Metastore server addresses, for example, **thrift://192.0.2.1:9083,thrift://192.0.2.2:9083**.)

myhive.hive.config.resources=/etc/hadoop/conf/core-site.xml,/etc/hadoop/conf/hdfs-site.xml (absolute path of the configuration files in **1**)

For details about other configurations, see **https://trino.io/docs/333/connector/hive.html#hive-configuration-properties**.

When you add a configuration, add the connector name as the prefix to the configuration name. In this example, the connector name is **myhive**.

Step 5 Save the configuration and restart the Presto service.

NOTE

Use the Presto client, JDBC, or UI tool to connect to the Presto service. Run the **show schemas from myhive;** command. If the database in the target cluster is displayed, the configuration is successful.

----End

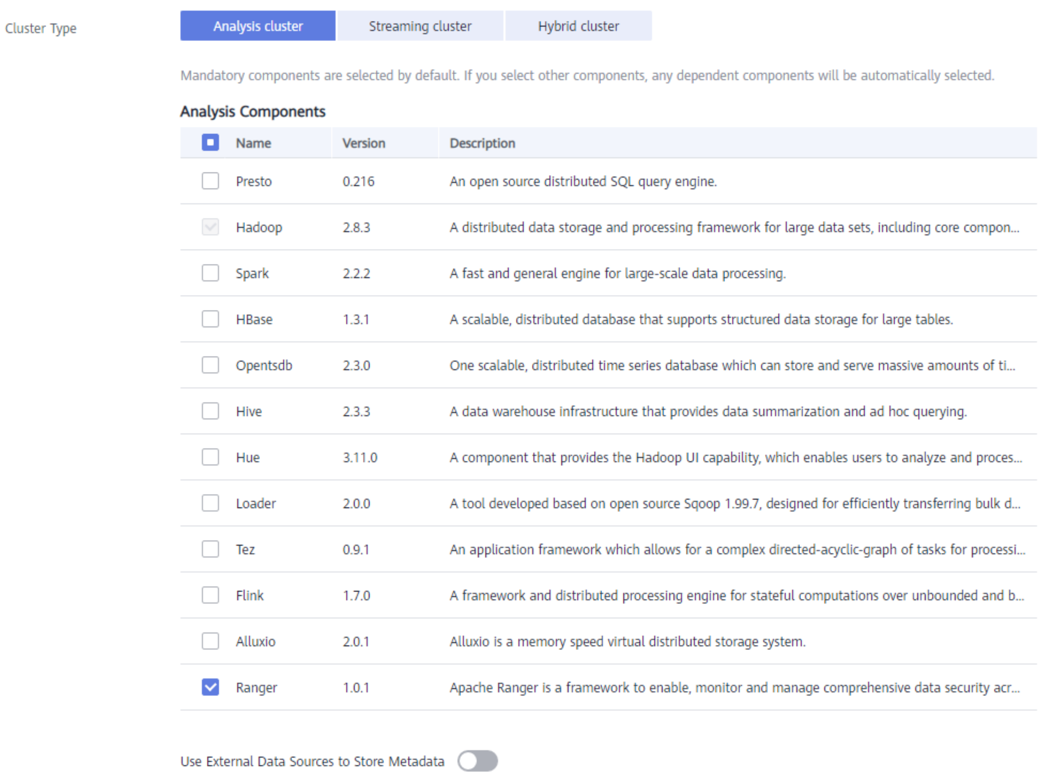
23 Using Ranger (MRS 1.9.2)

23.1 Creating a Ranger Cluster

Step 1 Create a cluster by referring to **Custom Purchase of a Cluster**. Select the Ranger component during cluster creation.

Currently, only normal MRS 1.9.2 clusters support Ranger. Security clusters with Kerberos authentication enabled do not support Ranger.

Figure 23-1 Selecting the Ranger component

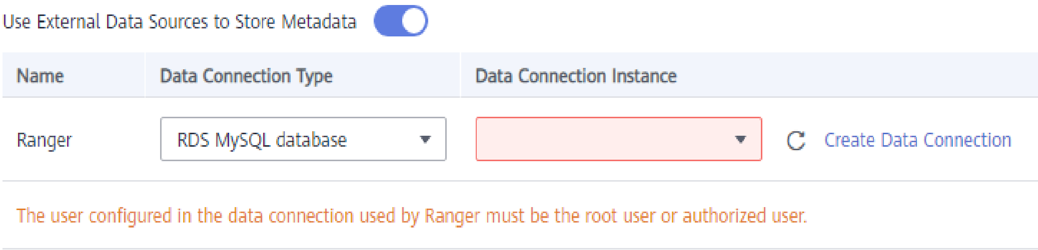


Step 2 Enable or disable **Use External Data Sources to Store Metadata**.

- Enabled: An external MySQL database is used to store the user, group, and policy data of Ranger.
- Disabled: The user, group, and policy data of Ranger is stored in the local database of the current cluster by default.

Step 3 If **Use External Data Sources to Store Metadata** is enabled, set **Data Connection Type** to **RDS MySQL database**. Select an existing data connection instance or click **Create Data Connection** to create a data connection.

Figure 23-2 Using the RDS MySQL database



NOTE

If the selected data connection is an **RDS MySQL database**, ensure that the database user is a **root** user. If the database user is not a **root** user, log in to the database as user **root** and run the following SQL statement to grant permissions to the database user. In the command, **${db_name}** and **${db_user}** indicate the database name and username entered during data connection creation.

```
grant select on mysql.user to ${db_user};
grant all privileges on ${db_name}.* to '${db_user}'@'%' with grant option;
grant reload on *.* to '${db_user}'@'%' with grant option;
flush privileges;
```

Step 4 Configure other parameters by referring to **Custom Purchase of a Cluster**.

NOTE

- After the cluster is created, Ranger does not control users' permissions to access Hive and HBase.
- When using Ranger to manage permissions for components such as Hive tables, submitting a Hive job on the console or client may result in a permission error message. To resolve this, adjust the user's database or table permissions in Ranger by following the policy addition procedure in **Configuring Hive/Impala Access Permissions in Ranger** or **Configuring HBase Access Permissions in Ranger**.

----End

23.2 Accessing the Ranger Web UI and Synchronizing Unix Users to the Ranger Web UI

You can manage Ranger on the Ranger web UI.

Accessing the Ranger Admin Web UI

Step 1 On the MRS management console, click the cluster name to go to the cluster details page.

Step 2 Click the **Components** tab.

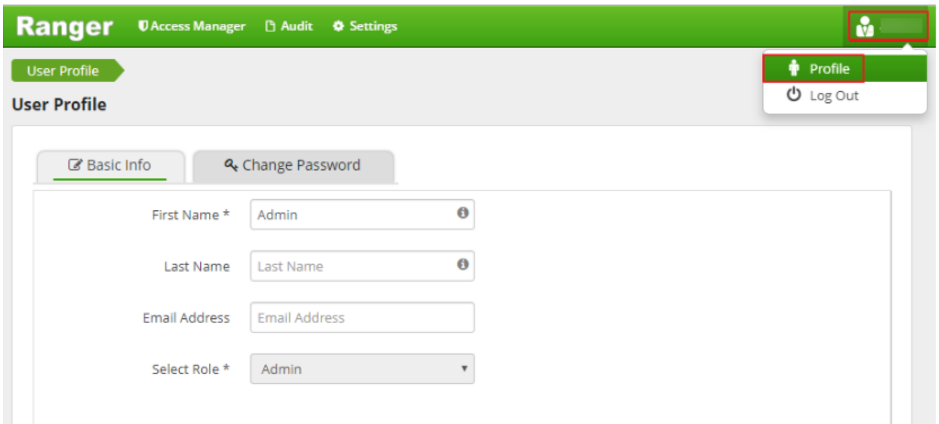
Step 3 Select **Ranger**. In **Ranger Summary**, click **RangerAdmin** corresponding to **Ranger Web UI**.

Step 4 On the Ranger web UI login page, the default username for MRS 1.9.2 is **admin** and the default password is **admin@12345**. The default username for MRS 1.9.3 is **admin** and the default password is **ranger@A1!**.

After logging in to the Ranger Web UI for the first time, change the password and keep it secure.

Step 5 Click the username in the upper right corner, choose **Profile** from the drop-down list, and click **Change Password** to change the password.

Figure 23-3 Changing the Ranger web UI login password



Step 6 After changing the password, click the username in the upper right corner, choose **Log Out** from the drop-down list, and use the new password to log in to the web UI again.

----End

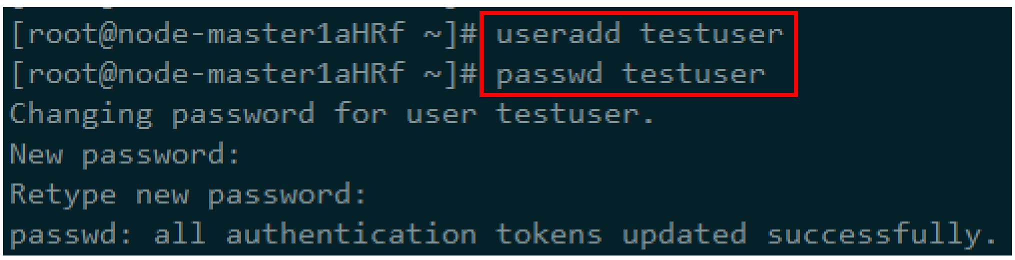
Using Ranger UserSync to Synchronize Unix OS Users on Cluster Nodes

Ranger UserSync is an important component of Ranger. It can synchronize Unix system users or LDAP users to the Ranger web UI. Currently, MRS can synchronize only Unix users on the node where the Ranger UserSync process resides.

Step 1 Log in to the node where the Ranger UserSync process is located.

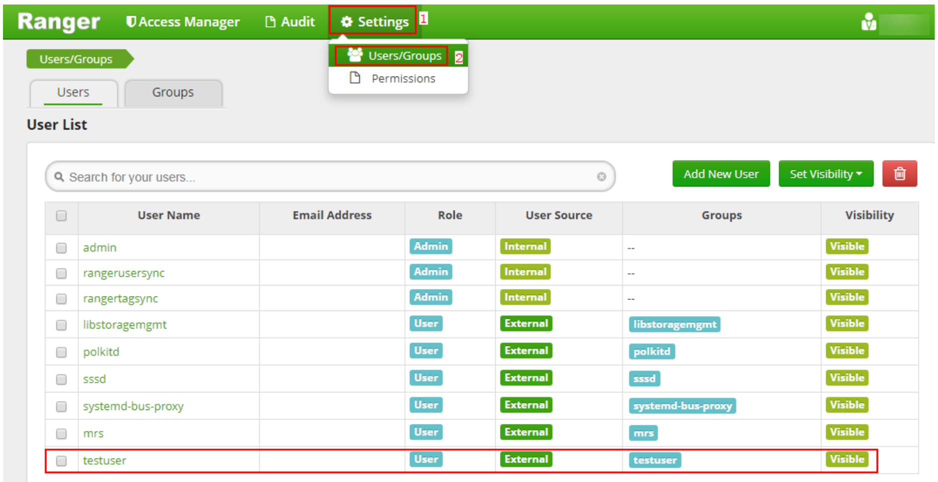
Step 2 Run the **useradd** command to add a system user, for example, **testuser**.

Figure 23-4 Adding the **testuser** system user



Step 3 After the user is added, wait for about 1 minute and log in to the Ranger web UI. Then, you can see that the user is synchronized.

Figure 23-5 User synchronization completed



----End

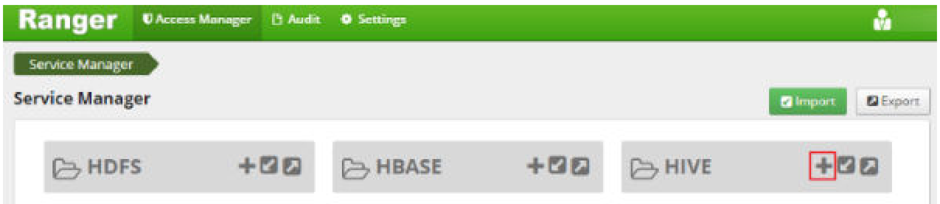
23.3 Configuring Hive/Impala Access Permissions in Ranger

After an MRS cluster with Ranger installed is created, Hive and Impala access control is not integrated into Ranger. This section describes how to integrate Hive into Ranger. Impala follows the same procedure.

Step 1 Log in to the Ranger web UI.

Step 2 In the **Service Manager** area, click + next to **HIVE** to add a Hive service.

Figure 23-6 Adding a Hive service



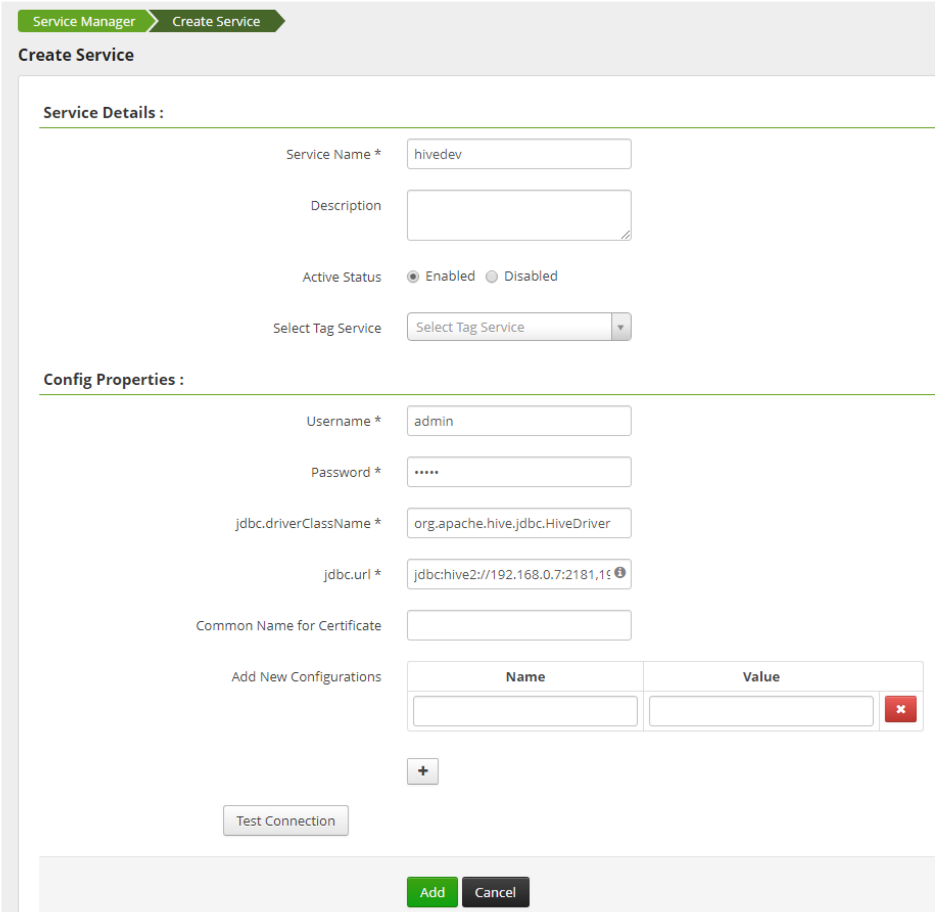
Step 3 Set the parameters for adding a Hive service according to **Table 23-1**. Use the default values for the parameters that are not listed in the table.

Table 23-1 Parameter description

Parameter	Description	Example Value
Service Name	Name of the service to be created. The value is fixed to **hivedev**.	hivedev

Parameter	Description	Example Value
Username	Set it based on the site requirements.	admin
Password	Set it based on the site requirements.	-
jdbc.driverClassName	Driver class for connecting to Hive. The value is fixed to **org.apache.hive.jdbc.HiveDriver**.	org.apache.hive.jdbc.HiveDriver
jdbc.url	URL for connecting to Hive. The format is ZooKeeper mode: jdbc:hive2://<host>:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2 **<host>** indicates a ZooKeeper address. To obtain the ZooKeeper address, log in to MRS Manager, choose **Services** > **ZooKeeper** > **Instance**, and view the management IP address of the ZooKeeper instance.	jdbc:hive2://xx.xx.xx.xx:2181,xx.xx.xx.xx:2181,xx.xx.xx.xx:2181/;serviceDiscovery-Mode=zooKeeper;zooKeeperNamespace=hiveserver2

Figure 23-7 Creating hivedev



Step 4 Click **Add** to add the service.

Step 5 Start the Ranger Hive plugin to authorize Ranger to manage Hive.

1. On the MRS management console, click the cluster name to go to the cluster details page.
2. Click the **Components** tab.
3. Choose **Hive** > **Service Configuration** and switch **Basic** to **All**.
4. Search for **hive.security.authorization** and modify the following configurations:
   – hive.security.authorization.enabled = true
   – hive.security.authorization.manager = org.apache.ranger.authorization.hive.authorizer.RangerHiveAuthorizerFactory
5. Click **Save Configuration** and select **Restart the affected services or instances** to restart the Hive service.

Step 6 Add an access control policy.

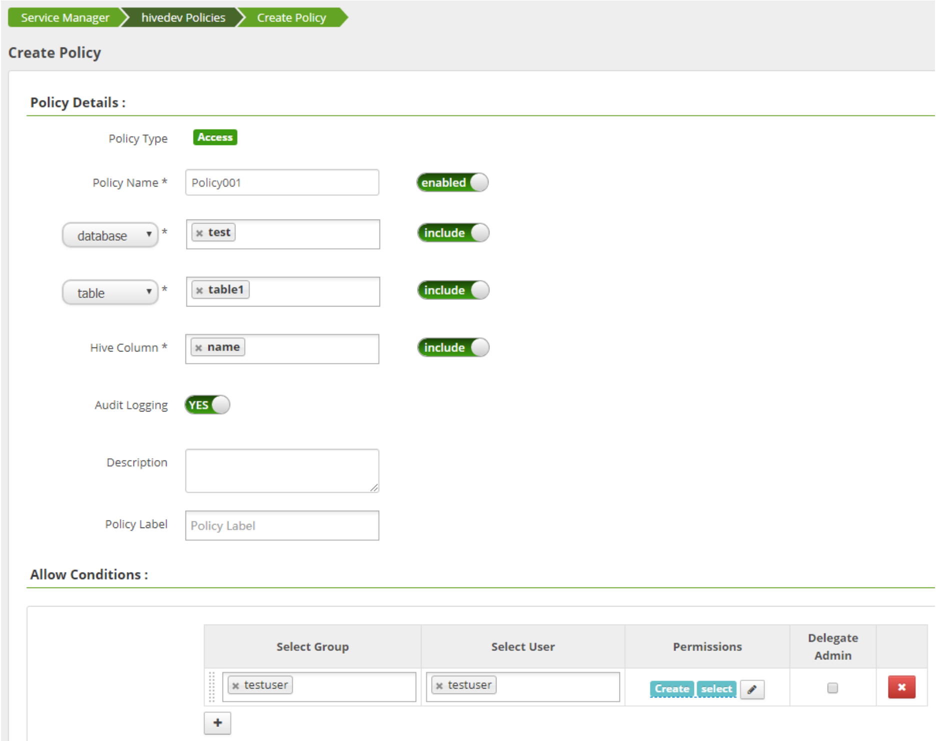
1. Log in to the Ranger web UI.
2. In the **HIVE** area, click the added service **hivedev**.
3. Click **Add New Policy** to add an access control policy.

4. Set the parameters according to **Table 23-2**. Use the default values for the parameters that are not listed in the table.

Table 23-2 Parameter description

Parameter	Description	Example Value
Policy Name	Policy name	Policy001
database	Name of the database that the policy allows to access	test
table	Name of the table corresponding to the database that the policy allows to access	table1
Hive Column	Column name of the table corresponding to the database that the policy allows to access	name
Allow Conditions	– **Select Group**: user group that the policy allows to access – **Select User**: user in the user group that the policy allows to access – **Permissions**: permissions that the policy allows the user to have	– Select Group: **testuser** – Select User: **testuser** – Permissions: **Create** and **Select**

Figure 23-8 Adding an access control policy for **hivedev**



5. Click **Add** to add the policy. According to the preceding policy, user **testuser** in the **testuser** user group has the **Create** and **Select** permissions on the **name** column of **table1** in the **test** database of Hive, but no permissions to access other columns.

Step 7 Log in to the Hive client by referring to **Using Hive for Data Analysis** and check whether Hive has been integrated into Ranger.

1. Log in to the node where the client is installed as the client installation user and run the following commands to go to Hive Beeline:

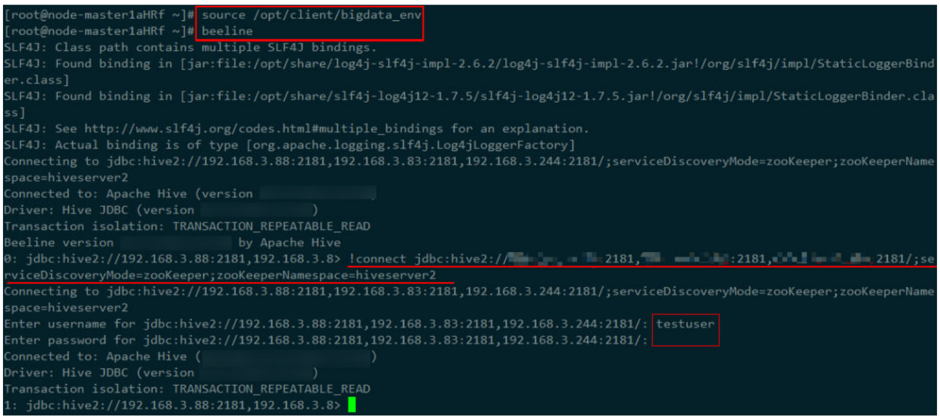
   source /opt/client/bigdata_env

   beeline

2. Run the following command to set up a connection and log in as user **testuser**:

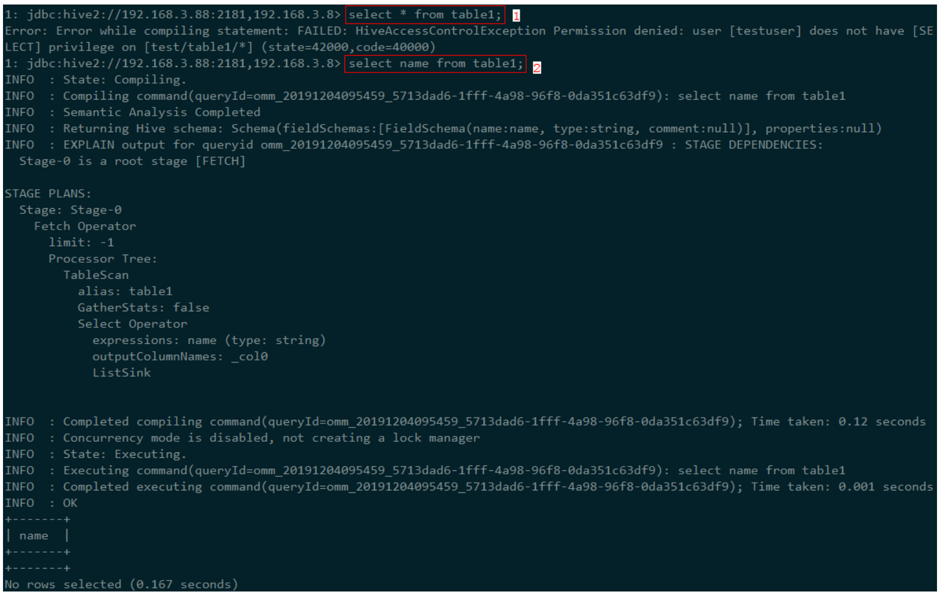
   !connect jdbc:hive2://xx.xx.xx.xx:2181,xx.xx.3.81:2181,192.168.3.153:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2

Figure 23-9 Logging in to Hive



3. Query data and check whether Ranger is integrated.

Figure 23-10 Verifying the integration of Ranger with Hive



----End

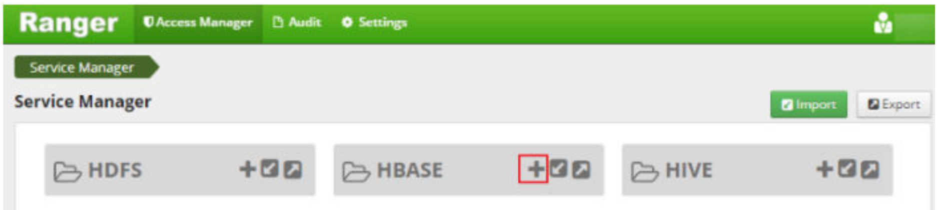
23.4 Configuring HBase Access Permissions in Ranger

After an MRS cluster with Ranger installed is created, HBase access control is not integrated into Ranger. This section describes how to integrate HBase into Ranger.

Step 1 Log in to the Ranger web UI.

Step 2 In the **Service Manager** area, click next to **HBASE** to add an HBase service.

Figure 23-11 Adding an HBase service

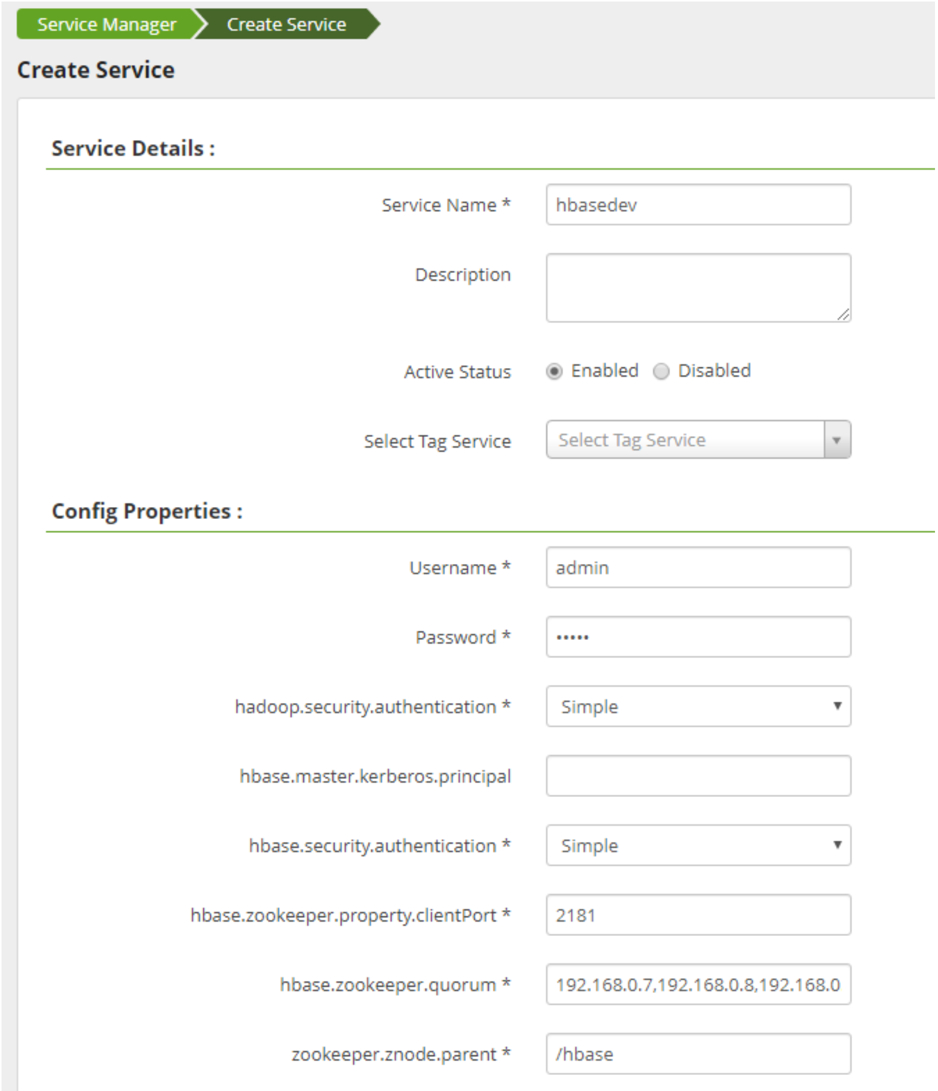


Step 3 Set the parameters for adding an HBase service according to **Table 23-3**. Use the default values for the parameters that are not listed in the table.

Table 23-3 Parameter description

Parameter	Description	Example Value
Service Name	Name of the service to be created. The value is fixed to **hbasedev**.	hbasedev
Username	Set it based on the site requirements.	admin
Password	Set it based on the site requirements.	-
hadoop.security.authentication	HBase authentication mode. The value is fixed to **Simple**.	Simple
hbase.security.authentication	HBase authentication mode. The value is fixed to **Simple**.	Simple
hbase.zookeeper.property.clientPort	Port number of ZooKeeper in the HBase cluster.	2181
hbase.zookeeper.quorum	ZooKeeper address in the HBase cluster.	192.168.0.7,192.168.0.8,192.168.0.9
zookeeper.znode.parent	Path of the root node of HBase in ZooKeeper. The value is fixed to **/hbase**.	/hbase

Figure 23-12 Creating hbasedev

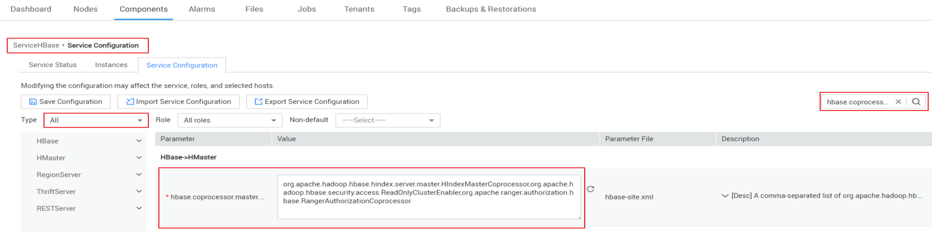


Step 4 Click **Add** to add the service.

Step 5 Start the Ranger HBase plugin to authorize Ranger to manage HBase.

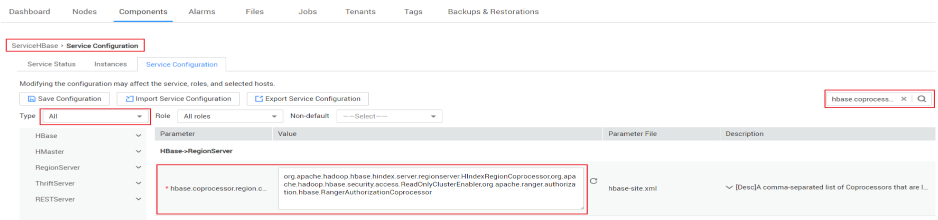
1. On the MRS management console, click the cluster name to go to the cluster details page.
2. Click the **Components** tab.
3. Choose **HBase** > **Service Configuration** and switch **Basic** to **All**.
4. Search for **hbase.security.authorization** and change its value to **true** (select the first HBase parameter).

Figure 23-13 Modifying **hbase.security.authorization**



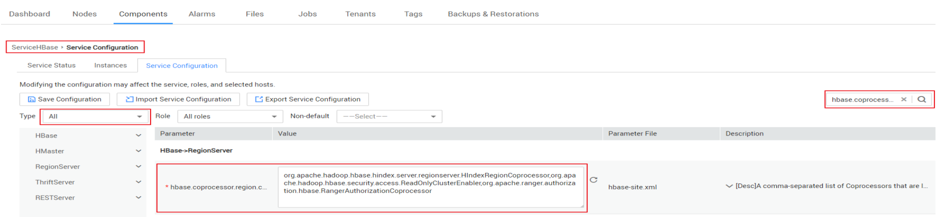
5. Search for **hbase.coprocessor.master.classes** and append **,org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor** to its original value.

Figure 23-14 hbase.coprocessor.master.classes



6. Search for **hbase.coprocessor.region.classes** and append **,org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor** to its original value.

Figure 23-15 hbase.coprocessor.region.classes



7. Click **Save Configuration** and select **Restart the affected services or instances** to restart the HMaster and RegionServer instances.

Step 6 Create a policy under **HBase Service hbasedev**.

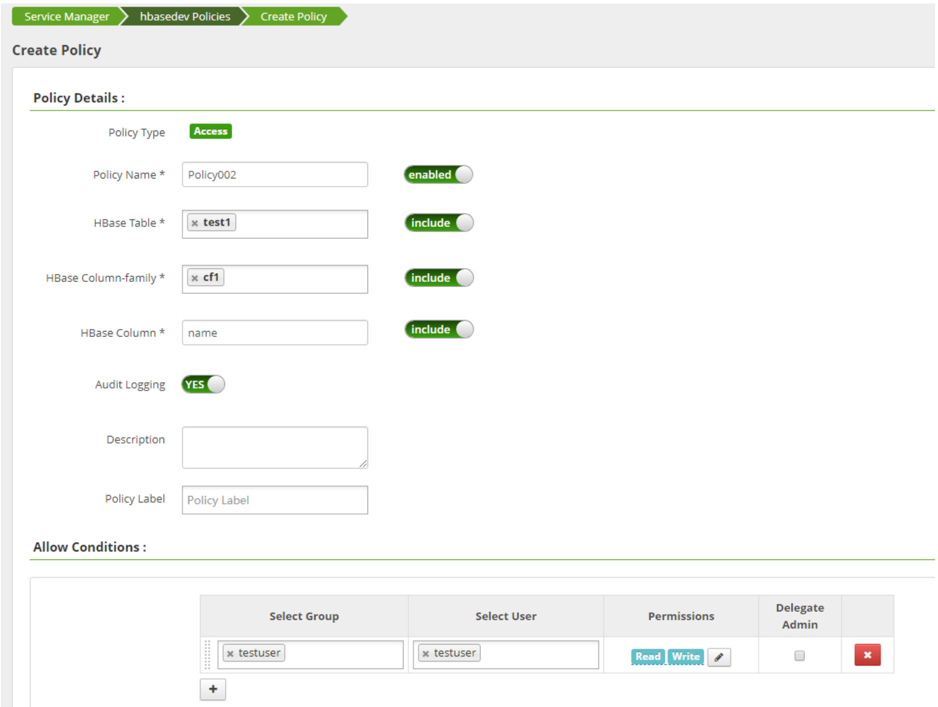
1. Log in to the Ranger web UI.
2. In the **HBASE** area, click the added service **hbasedev**.
3. Click **Add New Policy** to add an access control policy.
4. Set the parameters according to **Table 23-4**. Use the default values for the parameters that are not listed in the table.

Table 23-4 Parameter description

Parameter	Description	Example Value
Policy Name	Policy name	Policy002

Parameter	Description	Example Value
HBase Table	Name of the HBase table that the policy allows to access	test1
HBase Column-family	Column family of the HBase table that the policy allows to access	cf1
HBase Column	Column name of the table corresponding to the HBase table that the policy allows to access	name
Allow Conditions	– **Select Group**: user group that the policy allows to access – **Select User**: user in the user group that the policy allows to access – **Permissions**: permissions that the policy allows the user to have	– Select Group: **testuser** – Select User: **testuser** – Permissions: **Create** and **Select**

Figure 23-16 Adding an access control policy for **hbasedev**



5. Click **Add** to add the policy. According to the preceding policy, user **testuser** in the **testuser** user group has the **Create** and **Select** permissions on the **cf1:name** column in the **test1** table of the **default** namespace in HBase, but no permissions to access other columns.

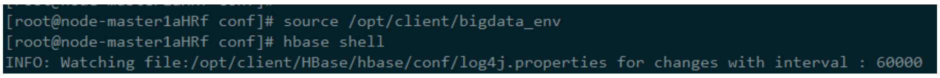
Step 7 Update and log in to the HBase client by referring to **Quickly Using HBase for Offline Data Analysis**, and check whether HBase has been integrated into Ranger.

1. Log in to the node where the client is installed as the client installation user and run the following commands to go to the HBase shell:

   source /opt/client/bigdata_env

   hbase shell

   Figure 23-17 Accessing the HBase shell

2. Add data and check whether Ranger is integrated.

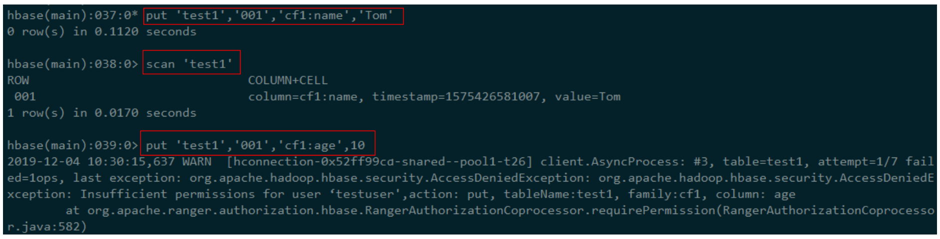
   a. Add data to the **cf1:name** column in the **test1** table.

      put 'test1','001','cf1:name','tom'

   b. Add data to the **cf1:age** column in the **test1** table. If the user has no permission to access this column, the data fails to be added.

      put 'test1','001','cf1:age',10

   Figure 23-18 Verifying the integration of Ranger with HBase

----End

24 Using Ranger (MRS 3.*x*)

24.1 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 management pages, 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 (MRS 3.x or Later)**. Choose **Cluster** > **Services** > **Ranger**. The Ranger service overview page is displayed.

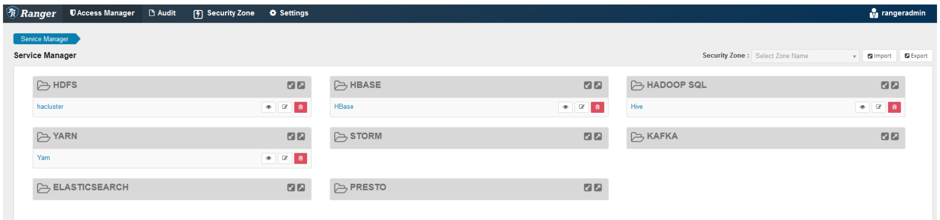
Step 2 Click **RangerAdmin** in the **Basic Information** area. The Ranger web UI is displayed.

- 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 24-1 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 (MRS 3.x or Later)**. 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.

For a normal cluster, 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 24-1**.

Table 24-1 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**.

Portal	Function
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**.

24.2 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

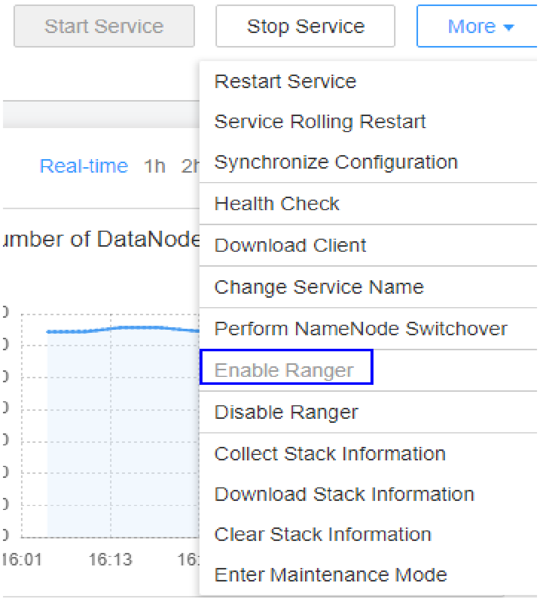
Step 1 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > **Services** > *Name of the service for which Ranger authentication is enabled*.

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 24-2**.
- 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 24-2 Enabling Ranger Authentication



Step 3 Perform a rolling service restart or restart the service.

----End

24.3 Adding a Ranger Permission Policy

In the newly installed MRS cluster, Ranger is installed by default, with the Ranger authentication model enabled. The Ranger administrator can set fine-grained security policies for accessing component resources through the component permission plug-ins.

Currently, the following components in a cluster in security mode support Ranger: HDFS, Yarn, HBase, Hive, Spark2x, Kafka, Storm..

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 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 Storm

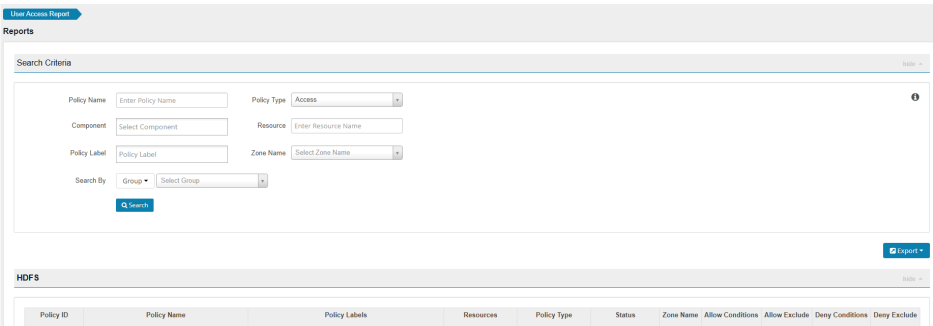
After the policies are added, wait for about 30 seconds for them to take effect.

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**.

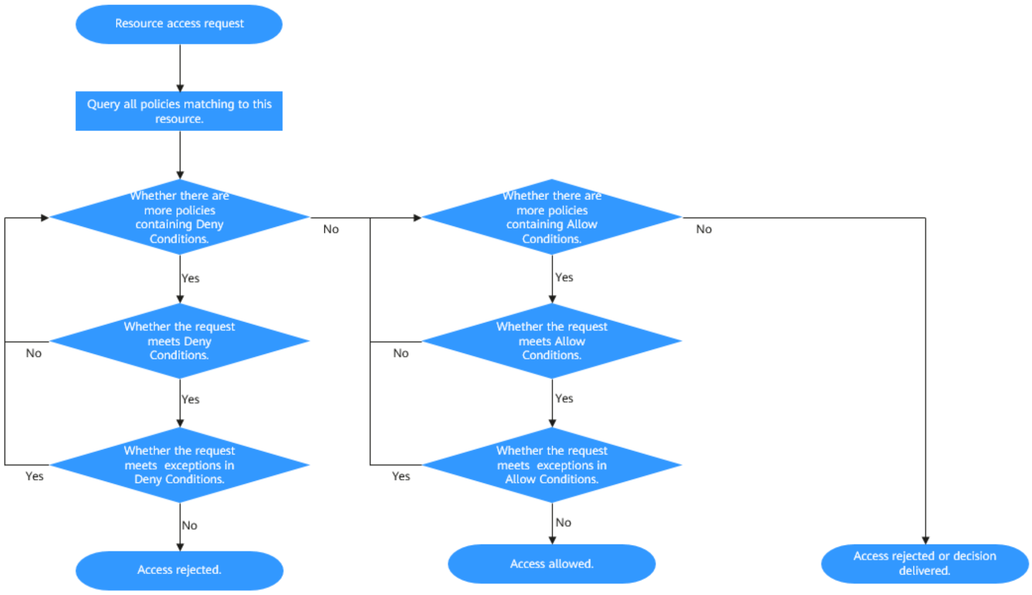
----End

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.

24.4 Configuration Examples for Ranger Permission Policy

24.4.1 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 24-2 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.
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 subdirectory has the policies of both the parent directory and the subdirectory. If the policy of the parent directory conflicts with that of the subdirectory, the policy of the subdirectory is used. ● **non-recursive**: recursion disabled ● **recursive**: recursion enabled
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 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.
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:

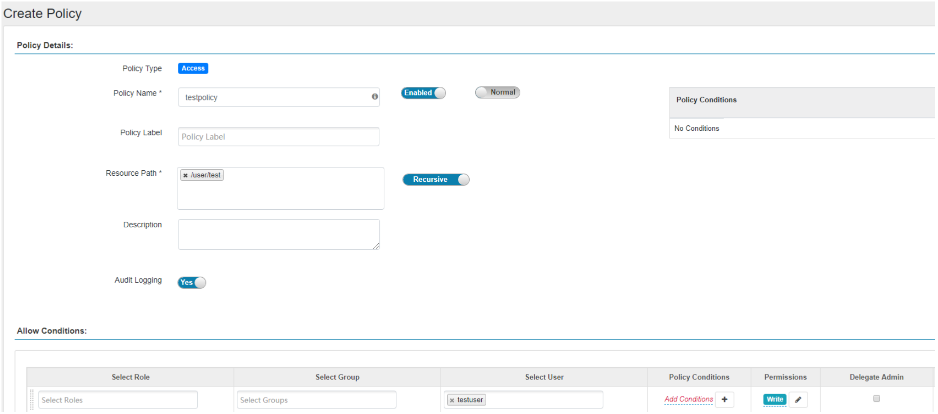


Table 24-3 Setting permissions

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**.
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**.

Task	Role Authorization
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

24.4.2 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.

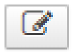
Step 4 Configure the parameters listed in the table below based on the service demands.

Table 24-4 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 **When an HBase Policy Is Added or Modified on Ranger, Wildcard Characters Cannot Be Used to Search for Existing HBase Tables**.
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.

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. ● **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.
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 24-5 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
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**, **incr** and **bulkload** 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.

Task	Role Authorization
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

24.4.3 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 24-6 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.
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.

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 24-7 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**.
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**.

Task	Role Authorization
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**.
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**.

Task	Role Authorization
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**.
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.
- The URL policy in the Ranger policy is involved in the scenario where the Hive table is stored on OBS. 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**.

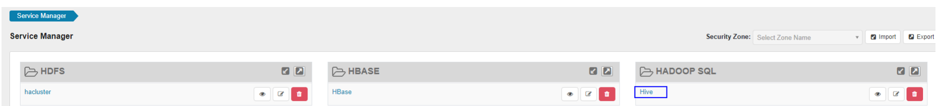
If a policy is no longer used, click  to delete it.

----End

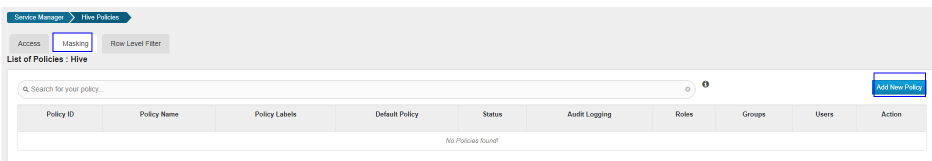
Hive Data Masking

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 24-8 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	Name of the Hive database to which the current policy applies.
Hive Table	Name of the Hive table to which the current policy applies.
Hive Column	Column name.
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

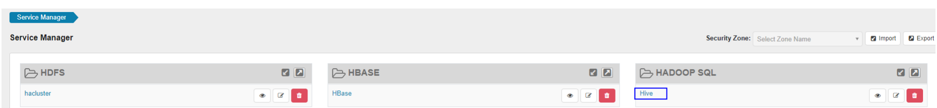
To process data, you must have the permission to submit tasks to the Yarn queue.

----End

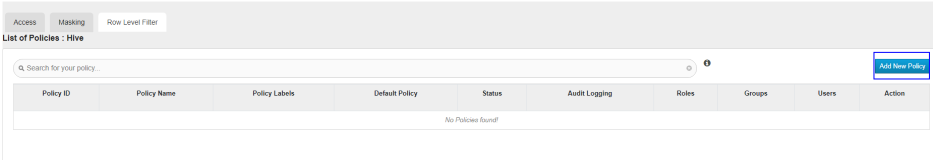
Hive Row-Level Data Filtering

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 24-9 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

24.4.4 Adding a Ranger Access Permission Policy for Impala

NOTE

This section applies only to MRS 3.1.5 or later.

Scenario

After the MRS cluster with Ranger installed is created, Hive/Impala permission control is integrated into Ranger. Impala reuses Hive permission policies. This section describes how to integrate Hive into Ranger.

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

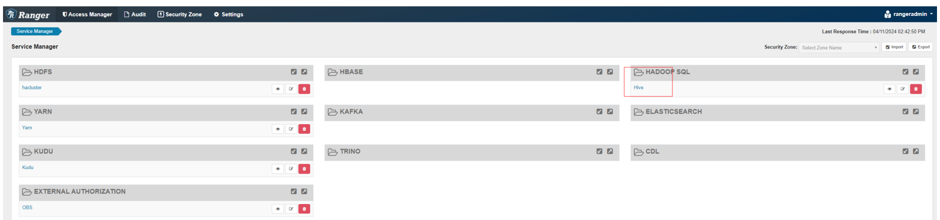
Step 1 For normal clusters with Kerberos authentication disabled, check whether Ranger authentication is enabled for Hive and Impala.

If Ranger authentication is not enabled, enable Ranger authentication for Hive, restart the Hive service, and then restart the Impala service. Enable Ranger authentication for Impala and restart the Impala service. By default, Ranger is enabled for security clusters with Kerberos authentication enabled. You can skip this step.

Step 2 For normal clusters, log in to FusionInsight Manager, choose **Cluster** > **Services** > **Ranger**, choose **Configurations** > **All Configurations**, and click **UserSync(Role)**. Add the following configuration parameter to the custom configuration item **ranger.usersync.config.expandor** and restart Ranger. Skip this step for security clusters with Kerberos authentication enabled.

Parameter	Value
ranger.usersync.sync.source	ldap

Step 3 Log in to the Ranger web UI as the Ranger administrator **rangeradmin**. For details, see **Logging In to the Ranger Web UI**. Select Hive.



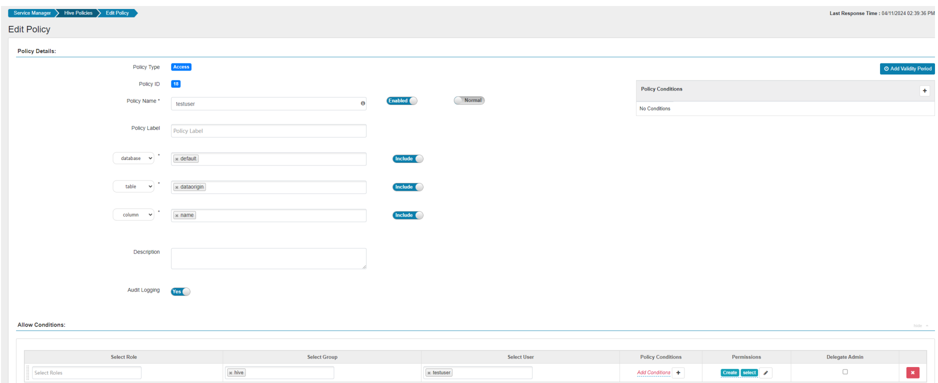
Step 4 Add an access control policy.

1. In the **HADOOP SQL** area, click the added service **Hive**.
2. Click **Add New Policy** to add an access control policy.
3. Set the parameters according to **Table 24-10**. Use the default values for the parameters that are not listed in the table.

Table 24-10 Parameters

Parameter	Description	Example Value
Policy Name	Policy Name	testuser
database	Name of the database that the policy allows to access	default
table	Name of the table corresponding to the database that the policy allows to access	dataorigin
Hive Column	Column name of the table corresponding to the database that the policy allows to access	name
Allow Conditions	– **Select Group**: user group that the policy allows to access – **Select User**: user in the user group that the policy allows to access – **Permissions**: permissions that the policy allows the user to have	– **Select Group**: **hive** – **Select User**: **testuser** – **Permissions**: **select** and **Create**

Figure 24-3 Adding the testuser access policy



4. Click **Add** to add the policy. According to the preceding policy, the **testuser** user in the **hive** user group has the **Create** and **Select** permissions on the **name** column of the **dataorigin** table in the **default** database of Hive, but no permissions to access other columns.

Step 5 Log in to the Impala client and check whether Ranger has been integrated with Impala.

1. Log in to the node where the client is installed as the client installation user and run the following command to initialize environment variables:

   source /opt/client/bigdata_env

2. Set up a connection and log in as user **testuser**.
   - For normal clusters with Kerberos authentication disabled, run the following command:

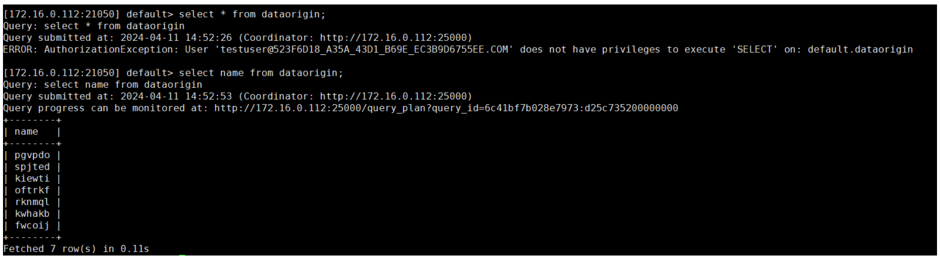
     impala-shell -i *<Impalad node IP address>* **-u** *testuser*
   - For security clusters with Kerberos authentication enabled, run the following command:

     kinit testuser

     Enter the password to log in.

     impala-shell -i *<Impalad node IP address>*

3. Query data and check whether Ranger is integrated.
   - Failed to run the **select * from dataorigin** command and an error message indicating insufficient permission is displayed.
   - The **select name from dataorigin** command is executed successfully and the preset permission is met.

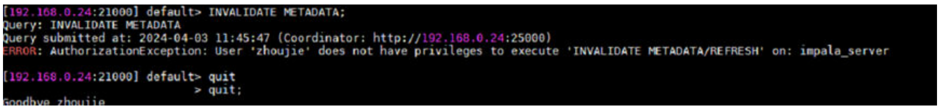


```
[172.16.0.112:21050] default> select * from dataorigin;
Query: select * from dataorigin
Query submitted at: 2024-04-11 14:52:26 (Coordinator: http://172.16.0.112:25000)
ERROR: AuthorizationException: User 'testuser@523F6D18_A35A_43D1_B69E_EC3B9D6755EE.COM' does not have privileges to execute 'SELECT' on: default.dataorigin

[172.16.0.112:21050] default> select name from dataorigin;
Query: select name from dataorigin
Query submitted at: 2024-04-11 14:52:53 (Coordinator: http://172.16.0.112:25000)
Query progress can be monitored at: http://172.16.0.112:25000/query_plan?query_id=6c41bf7b028e7973:d25c735200000000
+--------+
| name   |
+--------+
| pgvpdo |
| spjted |
| kiewti |
| oftrkf |
| rknmql |
| kwhakb |
| fwcoij |
+--------+
Fetched 7 row(s) in 0.11s
```

CAUTION

- If you have specified an HDFS path when running commands, you need to be assigned the read, write, and execution permissions on the HDFS path. 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.
- If a table is created in Hive, run the **invalidate metadata** command in Impala to update metadata. In this case, you need to grant the refresh permission to the user or run the **invalidate metadata** command as user **hive**. Otherwise, the following error message is displayed.



----End

24.4.5 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

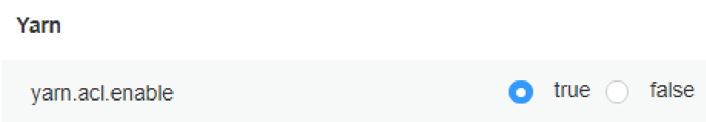
- The Ranger service has been installed and is running properly.
- You have created users, user groups, or roles for which you want to configure permissions.

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 24-4 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 24-11 Yarn 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.
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

Parameter	Description
Deny All Other Accesses	Whether to reject all other access requests. ● True: All other access requests are rejected. ● **False**: **Deny Conditions** can be configured.
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 24-12 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.

24.4.6 Adding a Ranger Access Permission Policy for Spark2x

Scenario

Ranger administrators can use Ranger to set permissions for Spark2x users.

NOTE

1. After Ranger authentication is enabled or disabled on Spark2x, you need to restart Spark2x.
2. Download the client again or manually update the client configuration file *Client installation directory*/**Spark2x/spark/conf/spark-defaults.conf**.

   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**
3. In Spark2x, spark-beeline (applications connected to JDBCServer) supports the Ranger IP address filtering policy (**Policy Conditions** in the Ranger permission policy), while spark-submit and spark-sql do not.

Prerequisites

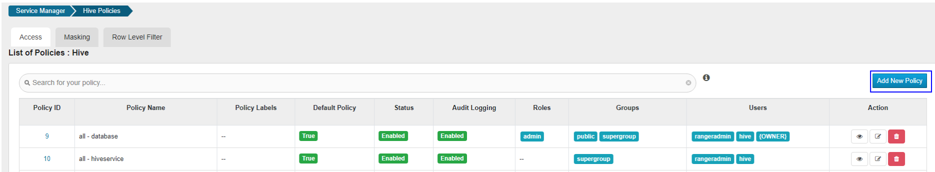
- The Ranger service has been installed and is running properly.
- Ranger authentication of Hive has been enabled and the Spark Ranger authentication function is reactivated following the sequential reboot of Hive and then Spark. The Spark service is restarted after the Spark Ranger authentication function 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 24-13 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 24-14 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 Spark2x client installation directory is **/opt/client**, run **source /opt/client/bigdata_env**. 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**.

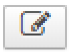
Task	Operation
VIEW and **INDEX** permissions	1. Enter the policy name in **Policy Name**. 2. On the right side of **database**, enter the database name and select the corresponding database. (If you want to delete a database, enter the database name and select *.) On the right side of **table**, enter 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.
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.

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/spark-defaults.conf** directory, 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 24-15 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/spark-defaults.conf** directory, 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 24-16 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 Spark2x database to which the current policy applies.
Hive Table	Name of the Spark2x 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 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

24.4.7 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 24-17 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 24-5 Kafka permission parameters

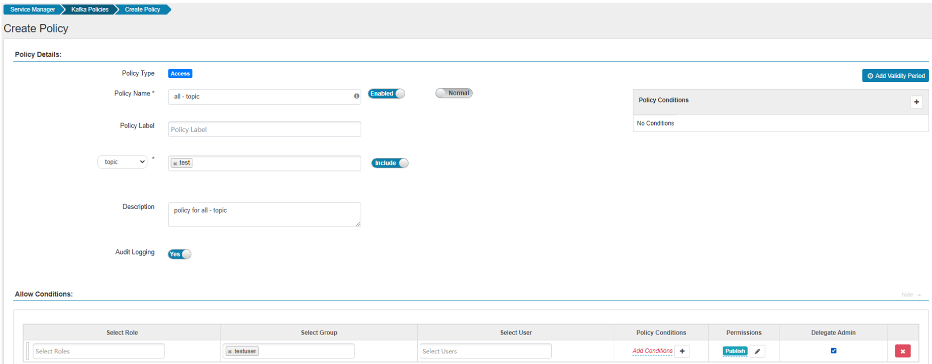
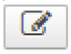


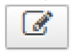
Table 24-18 Setting permissions

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.

<table>
<tr><th>Scenario</th><th>Role Authorization</th></tr>
<tr><td>Setting the permission for a user to delete a topic</td><td>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.</td></tr>
<tr><td>Setting the permission for a user to query a topic</td><td>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.</td></tr>
<tr><td>Setting the production permission of a user on a topic</td><td>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.</td></tr>
<tr><td>Setting the consumption permission of a user on a topic</td><td>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.</td></tr>
<tr><td>Setting the permission for a user to expand a topic (by adding partitions)</td><td>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.</td></tr>
</table>

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.)

<table>
<tr><th>Scenario</th><th>Role Authorization</th></tr>
<tr><td>Setting the permission for a user to query the cluster configuration</td><td>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.</td></tr>
<tr><td>Setting the Idempotent Write permission in a cluster for a user</td><td>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.</td></tr>
<tr><td>Setting the permission to migrate partitions in a cluster for a user</td><td>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).</td></tr>
</table>

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 Kakfa 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

24.4.8 Adding a Ranger Access Permission Policy for Storm

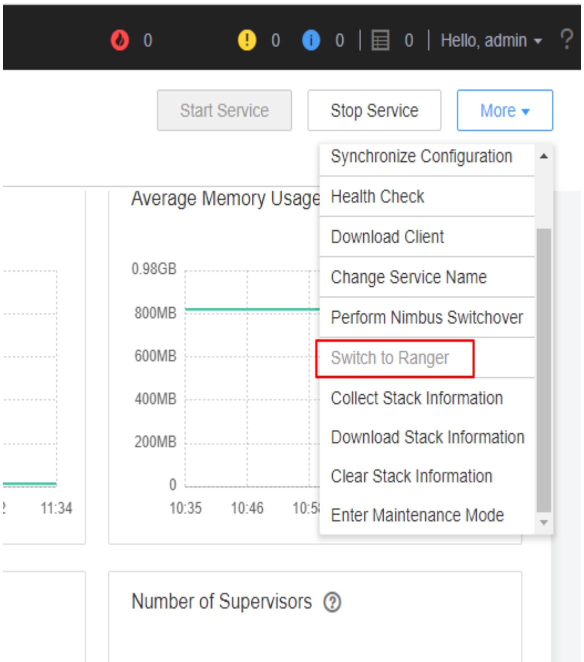
Scenario

Ranger administrators can use Ranger to set permissions for Storm 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 Ranger authentication function has been enabled on the page. The option in the following figure controls whether to enable the Ranger plug-in for permission control. If the function is enabled, the Ranger authentication is used. Otherwise, the authentication mechanism of the component is used.

Figure 24-6 Enabling Ranger authentication



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 **Storm** in the **STORM** area.

Step 3 Click **Add New Policy** to add a Storm permission control policy.

Step 4 Configure the parameters listed in the table below based on the service demands.

Table 24-19 Storm permission parameters

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 Name	Policy name, which can be customized and must be unique in the service. The **include** policy applies to the current input object, and the **exclude** policy applies to objects other than the current input object.
Policy Label	A label specified for the current policy. You can search for reports and filter policies based on labels.

Parameter	Description
Storm Topology	Name of the topology to which the current policy applies. One or more values can be entered.
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 permissions. ● **Submit Topology**: Submit a topology. **NOTE** The Submit Topology permission takes effect only when **Storm Topology** is set to *. ● **File Upload**: Upload a file. ● **File Download**: Download a file. ● **Kill Topology**: Delete a topology. ● **Rebalance**: Perform the rebalance operation. ● **Activate**: Activate the topology permission. ● **Deactivate**: Deactivate the topology permission. ● **Get Topology Conf**: Obtain topology configurations. ● **Get Topology**: Obtain a topology. ● **Get User Topology**: Obtain user's topology. ● **Get Topology Info**: Obtain topology information. ● **Upload New Credential**: Upload a new credential. ● **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**.

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

24.5 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 24-20**. If there are a large number of audit records, you can filter them in the search box by keyword.

Table 24-20 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.

Tab	Description
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

24.6 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**.

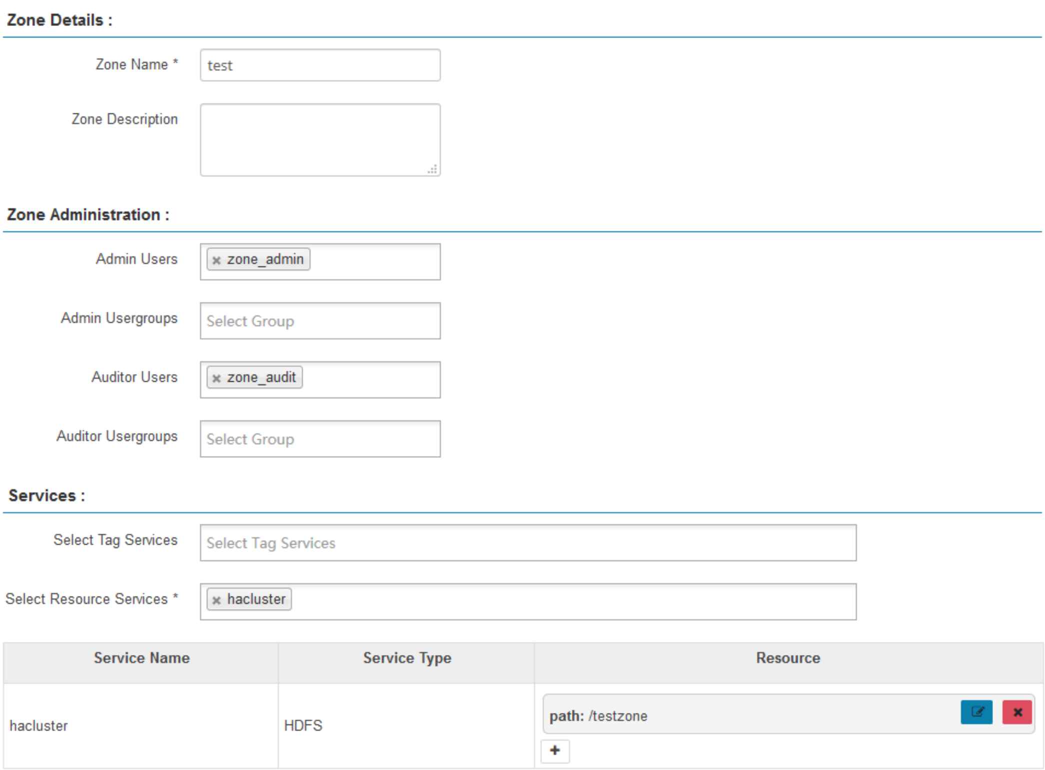
Step 2 Click **Security Zone**. On the zone list page, click  to add a zone.

Table 24-21 Parameters for configuring a security zone

Parameter	Description	Example Value
Zone Name	Security zone	test
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

Parameter	Description	Example Value
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.

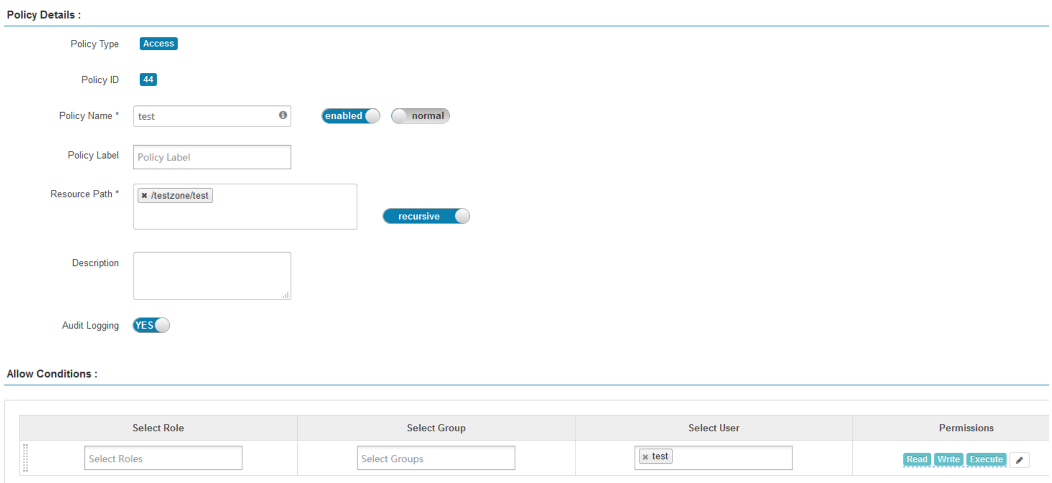
Step 3 Click the permission plug-in name of a component. The page for security access policy list of the component is displayed.

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 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 Storm**

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

24.7 Changing the Ranger Data Source to LDAP for a Normal Cluster

By default, the Ranger data source of the security cluster can be accessed by FusionInsight Manager LDAP users. By default, the Ranger data source of a common cluster can be accessed by Unix users.

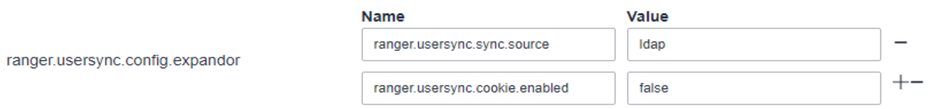
Prerequisites

- The cluster is in normal mode.
- The Ranger component has been installed.

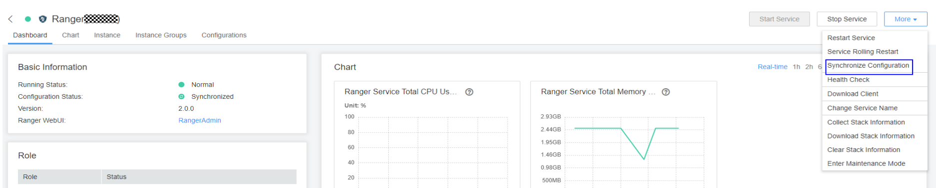
Procedure

Step 1 Log in to FusionInsight Manager by referring to **Accessing FusionInsight Manager (MRS 3.x or Later)**, choose **Cluster** > **Services** > **Ranger**, click **Configurations**, and click **All Configurations**. Click **UserSync(Role)** and click **Customization**.

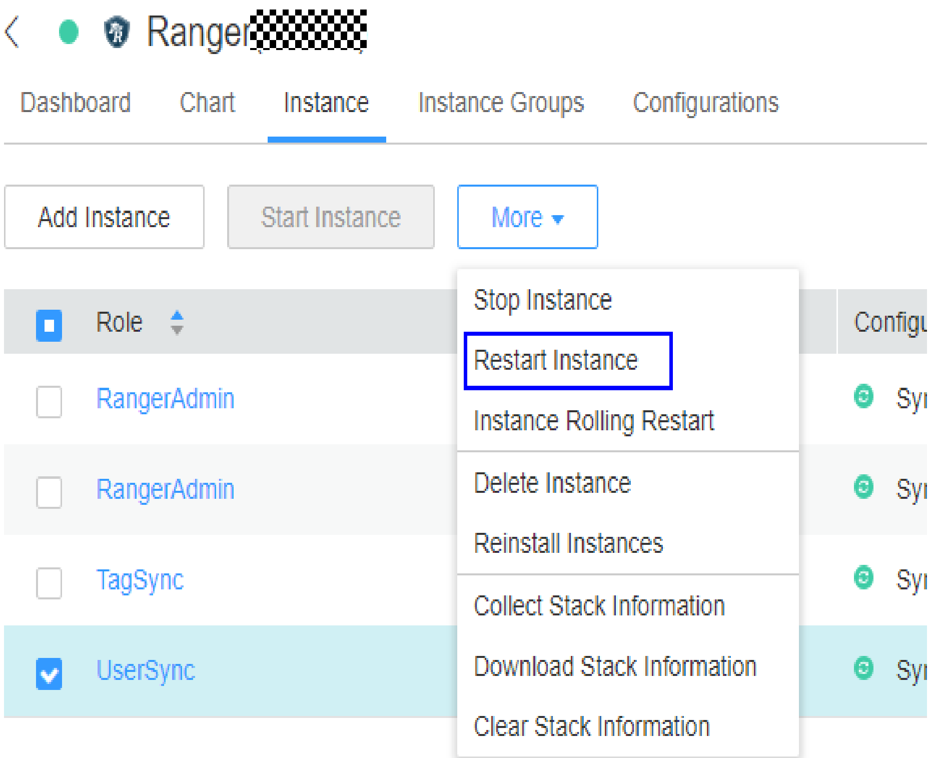
Step 2 In the **ranger.usersync.config.expandor** area, set **ranger.usersync.sync.source** to **ldap** and **ranger.usersync.cookie.enabled** to **false**, as shown in the following figure.



Step 3 In the upper right corner of the Ranger **Dashboard** page, click **More** and choose **Synchronize Configuration**.



Step 4 On the Ranger instance page, select the **UserSync** instance and choose **More** > **Restart Instance**.



Step 5 On the **Dashboard** page of the Ranger service, click **RangerAdmin** and choose **Settings** > **Users/Groups/Roles** to check whether LDAP users exist.

----End

24.8 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.

----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.

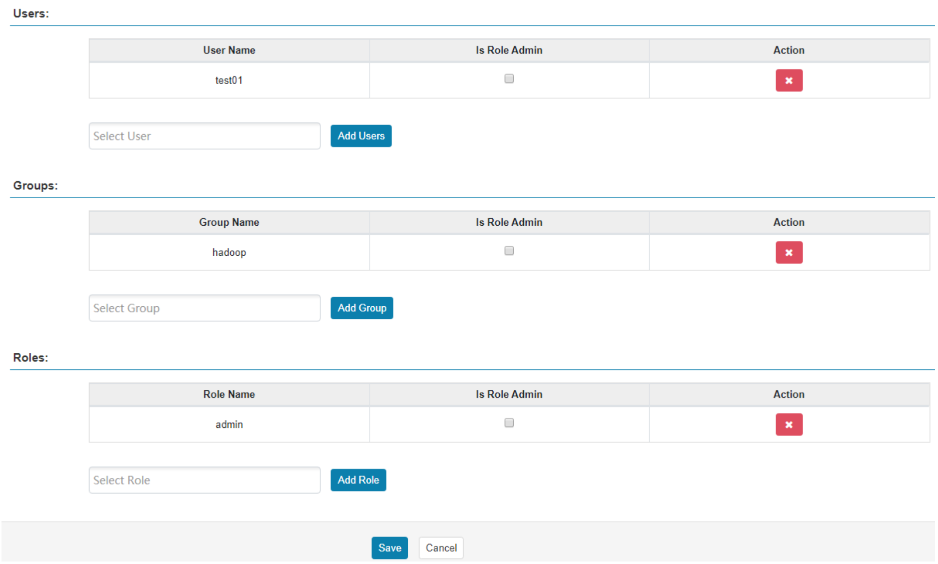
Step 1 Log in to the Ranger management page.

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

24.9 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)
- TagSync: /var/log/Bigdata/ranger/tagsync (run logs)
- UserSync: /var/log/Bigdata/ranger/usersync (run logs)

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 24-22 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

Log Levels

Table 24-23 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 24-23 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 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 24-24 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

24.10 Common Issues About Ranger

24.10.1 Why Ranger Startup Fails During the Cluster Installation?

Problem

During cluster installation, Ranger fails to be started, and the error message "ERROR: cannot drop sequence X_POLICY_REF_ACCESS_TYPE_SEQ " is displayed in the task list of the Manager process. How do I resolve this problem and properly install Ranger?

Answer

This issue may occur when two RangerAmdin instances are installed. If the instance installation fails, manually restart one RangerAdmin instance and then restart the other instance.

24.10.2 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.

24.10.3 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.

If Kerberos authentication is disabled, only the **admin** user can log in to the Ranger page because Ranger does not synchronize user data from Manager.

24.10.4 When an HBase Policy Is Added or Modified on Ranger, Wildcard Characters Cannot Be Used to Search for Existing HBase Tables

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)
```

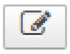


```
at
org.apache.hbase.thirdparty.io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.java:362)
```

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

24.10.5 Why Can't I View the Created MRS User on the Ranger Management Page?

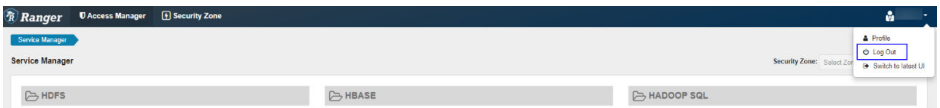
Question

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 Ranger web UI.



2. Log in to the system as user **rangeradmin** (default password: **Rangeradmin@123**) or another user who has Ranger administrator permission.

24.10.6 What Should I Do If MRS Users Failed to Be Synchronized to the Ranger Web UI?

Question

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**.

25 Using Spark (for Versions Earlier Than MRS 3.x)

25.1 Getting Started with Spark

This section describes how to use Spark to submit a SparkPi job. SparkPi, a typical Spark job, is used to calculate the value of Pi (π).

Procedure

Step 1 Prepare the SparkPi program.

Multiple open source Spark sample programs are provided, including SparkPi. Click **https://archive.apache.org/dist/spark/spark-2.1.0/spark-2.1.0-bin-hadoop2.7.tgz** to download the Spark program package.

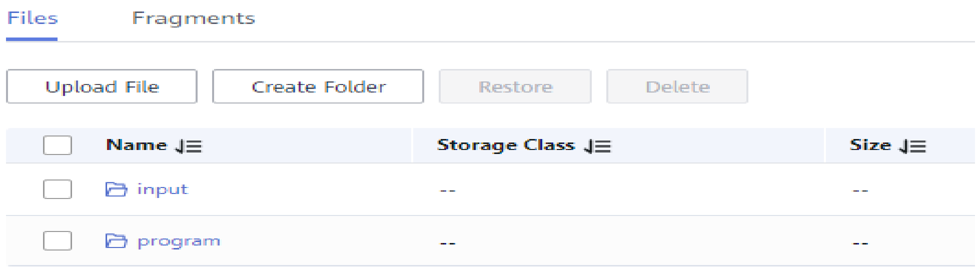
Decompress the software package to obtain the **spark-examples_2.11-2.1.0.jar** file, the sample program package, in the **spark-2.1.0-bin-hadoop2.7/examples/jars** directory. The **spark-examples_2.11-2.1.0.jar** sample program package contains the SparkPi program.

Step 2 Upload data to OBS.

1. Log in to OBS Console.
2. Choose **Parallel File System** > **Create Parallel File System** to create a file system named **sparkpi**.

   sparkpi is only an example. The file system name must be globally unique. Otherwise, the parallel file system fails to be created. Use the default values for other parameters.
3. Click the file system name **sparkpi** and click **Files**.
4. Click **Create Folder** to create the **program** folder. **Figure 25-1** shows the folder.

Figure 25-1 Folder list



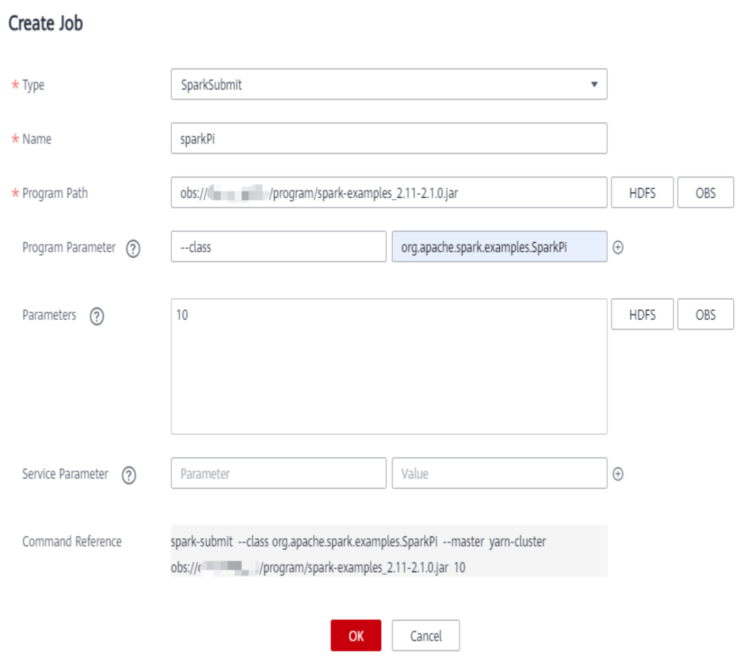
5. Go to the **program** folder, click **Upload Object**, select the program package downloaded in **Step 1** from the local PC, and set **Storage Class** to **Standard**.

Step 3 Log in to the MRS console. On the displayed **Active Clusters** page, click the name of the desired cluster in the cluster list.

Step 4 Submit the SparkPi job.

On the details page of the MRS cluster, click the **Jobs** tab. On this tab page, click **Create**. The **Create Job** dialog box is displayed. For details, see **Running a Spark Job**.

Figure 25-2 sparkPi job



- Set **Type** to **SparkSubmit**.
- Set **Name** to **sparkPi**.
- Set **Program Path** to the path where programs are stored on OBS, for example, **obs://sparkpi/program/spark-examples_2.11-2.1.0.jar**.
- In **Program Parameter**, select **--class** for **Parameter** and set **Value** to **org.apache.spark.examples.SparkPi**.

- Set **Parameters** to **10**.
- Leave **Service Parameter** 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 5 View the job execution result.

1. Go to the **Jobs** tab page and view job execution status.

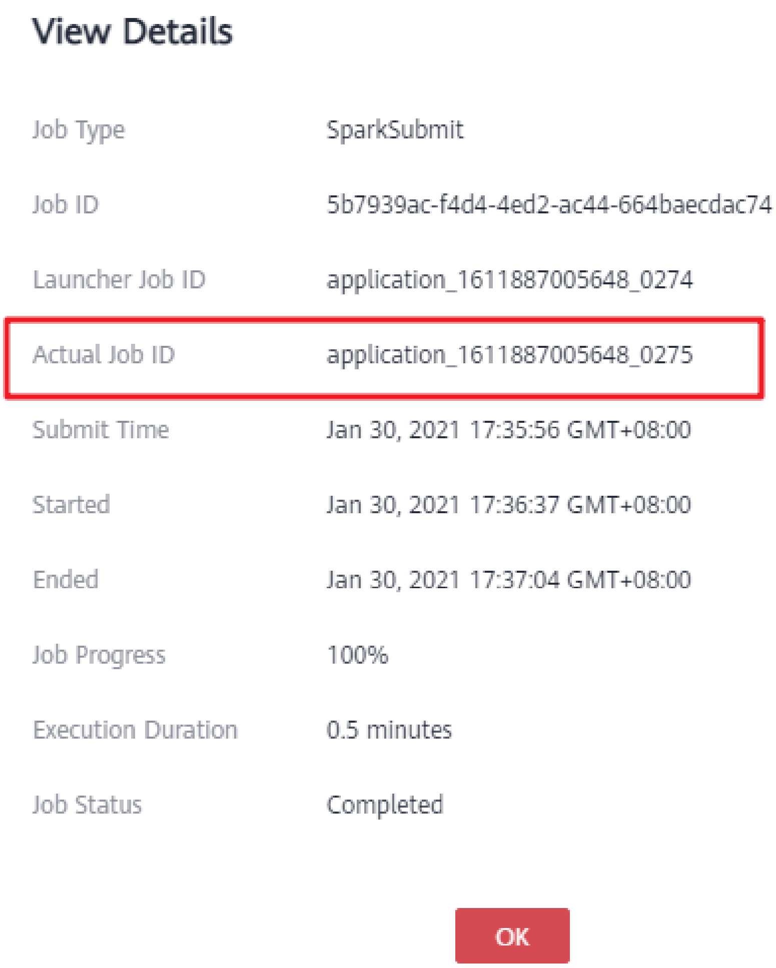
   The job execution takes a while. After the jobs are complete, refresh the job list.

   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. Go to the native Yarn page and view the job output information.

   a. On the **Jobs** tab page, locate the row that contains the target job and click **View Details** in the **Operation** column to obtain the actual job ID.

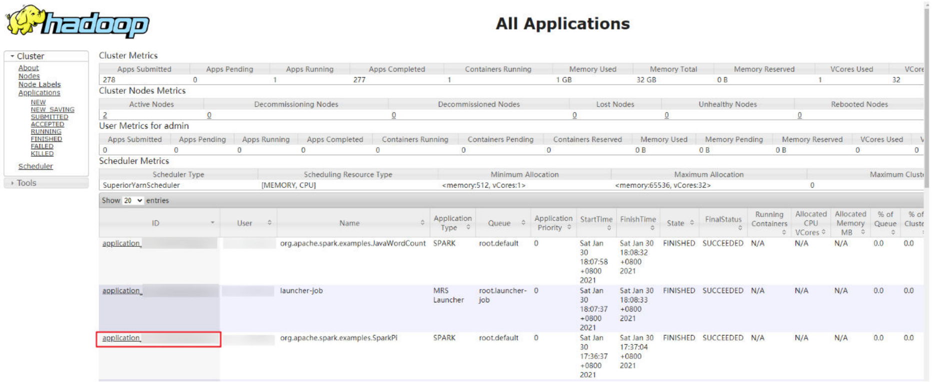
   Figure 25-3 Actual job ID



   b. Log in to Manager and choose **Services** > **Yarn** > **ResourceManager WebUI** > **ResourceManager (Active)**. The Yarn page is displayed.

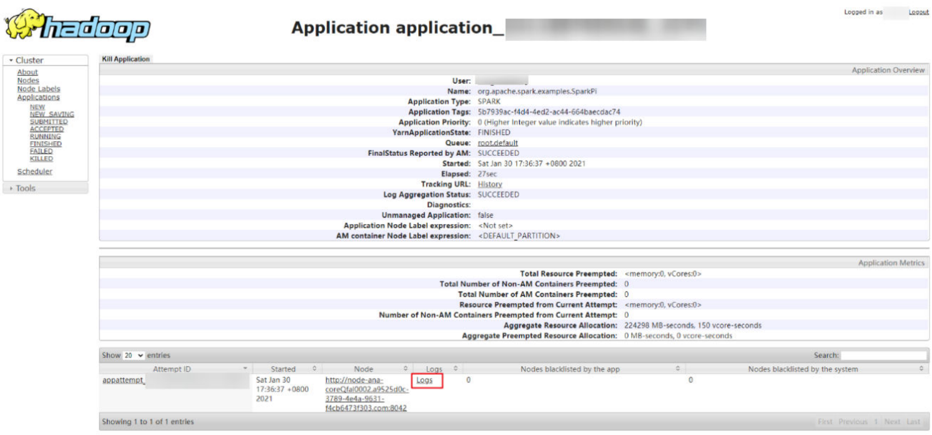
c. Click the ID corresponding to the actual job ID.

Figure 25-4 Yarn web UI



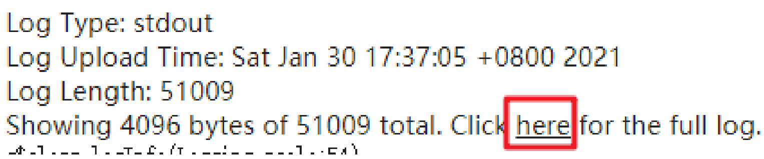
d. Click **Logs** in the job log area.

Figure 25-5 sparkPi job logs



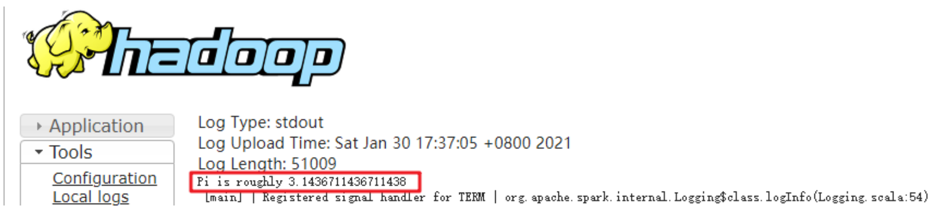
e. Click **here** to obtain more detailed logs.

Figure 25-6 More detailed logs of sparkPi jobs



f. Obtain the job execution result.

Figure 25-7 sparkPi job execution result



----End

25.2 Getting Started with Spark SQL

Spark provides the Spark SQL language that is similar to SQL to perform operations on structured data. This section describes how to use Spark SQL from scratch. Create a table named **src_data**, write a data record in each row of the table, and store the data in the **mrs_20160907** cluster. Then use SQL statements to query data in the table, and delete the table at last.

Prerequisites

You have obtained the AK/SK for writing data from an OBS data source to a Spark SQL table. To obtain it, perform as follows:

1. Log in to the management console.
2. Click the username and select **My Credentials** from the drop-down list.
3. On the displayed **My Credentials** page, click **Access Keys**.
4. Click **Create Access Key** to switch to the **Create Access Key** dialog box.
5. Enter the password and SMS verification code, and click **OK** to download the access key. Keep the access key secure.

Procedure

Step 1 Prepare data sources for Spark SQL analysis.

The sample text file is as follows:

```
abcd3ghji
efgh658ko
1234jjyu9
7h8kodfg1
kk99icxz3
```

Step 2 Upload data to OBS.

1. Log in to OBS Console.
2. Choose **Parallel File System** > **Create Parallel File System** to create a file system named **sparksql**.

   sparksql is only an example. The file system name must be globally unique. Otherwise, the parallel file system fails to be created.
3. Click the name of the **sparksql** file system and click **Files**.
4. Click **Create Folder** to create the **input** folder.
5. Go to the **input** folder, choose **Upload File** > **add file**, select the local TXT file, and click **Upload**.

Step 3 Log in to the MRS console. On the displayed **Active Clusters** page, click the name of the desired cluster in the cluster list.

Step 4 Import the text file from OBS to HDFS.

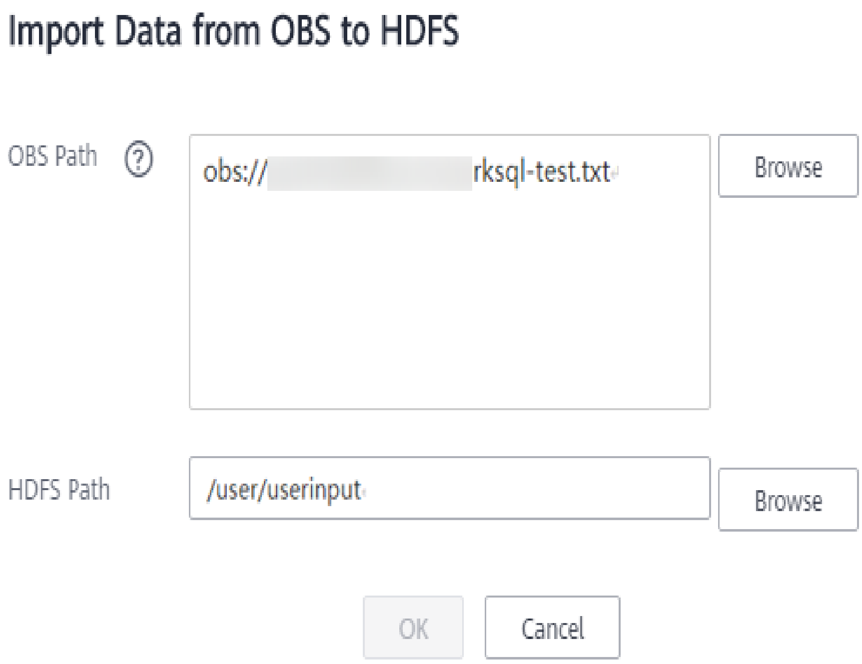
1. Click the **Files** tab.
2. On the **HDFS File List** tab page, click **Create Folder**, and create a folder named **userinput**.

3. Go to the **userinput** folder, and click **Import Data**.
4. Select the OBS and HDFS paths and click **OK**.

   OBS Path: **obs://sparksql/input/sparksql-test.txt**

   HDFS Path: **/user/userinput**

   Figure 25-8 Importing data from OBS to HDFS



Step 5 Submit the SQL statement.

1. On the details page of the MRS cluster, click the **Jobs** tab. For details, see **Running a Spark Job**.

   A job can be submitted only when the **mrs_20160907** cluster is in the **Running** state.
2. Enter the Spark SQL statement for table creation.

   When entering Spark SQL statements, ensure that the statement characters are not more than 10,000.

   Syntax:

   CREATE [EXTERNAL] **TABLE** [IF NOT EXISTS] *table_name* [(col_name data_type [COMMENT col_comment], ...)] [COMMENT table_comment] [PARTITIONED **BY** (col_name data_type [COMMENT col_comment], ...)] [CLUSTERED **BY** (col_name, col_name, ...) [SORTED **BY** (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS] [ROW FORMAT row_format] [STORED **AS** file_format] [LOCATION hdfs_path];

   You can use the following two methods to create a table example:

   – Method 1: Create table **src_data** and write data in every row.

     ▪ The data source is stored in the **/user/omm/userinput** folder of HDFS: ***create external table*** *src_data* ***(line string) row format delimited fields terminated by '\\n' stored as textfile location*** *'/user/omm/userinput'*;

- The data source is stored in the **/sparksql/input** folder of OBS:
  create external table *src_data***(line string) row format delimited fields terminated by '\\n' stored as textfile location** '*obs://AK:SK@sparksql/input*';

  For details about how to obtain the AK/SK, see **Prerequisites**.

– Method 2: Create table **src_data1** and load data to the table in batches.

  create table *src_data1* **(line string) row format delimited fields terminated by ','** ;

  load data inpath '*/user/omm/userinput/sparksql-test.txt*' **into table** *src_data1*;

NOTE

When method 2 is used, the data from OBS cannot be loaded to the created tables directly.

3. Enter the Spark SQL statement for table query.

   Syntax:

   SELECT col_name **FROM** *table_name*;

   Example of querying all data in the **src_data** table:

   select * from src_data;

4. Enter the Spark SQL statement for table deletion.

   Syntax:

   DROP TABLE [IF EXISTS] *table_name*;

   Example of deleting the **src_data** table:

   drop table src_data;

5. Click **Check** to check the statement correctness.
6. Click **OK**.

   After the Spark SQL statements are submitted, the statement execution results are displayed in the result column.

Step 6 Delete the cluster.

----End

25.3 Using the Spark Client

After an MRS cluster is created, you can create and submit jobs on the client. The client can be installed on nodes inside or outside the cluster.

- Node in the cluster: After an MRS cluster is created, the client has been installed on the master and core nodes in the cluster by default. For details about how to log in to a node where the client has been installed, see **Installing a Client on a Node Inside a Cluster**.
- Node outside the cluster: You can install the client on a node outside the cluster. For details about how to log in to a node where the client has been installed, see **Installing a Client on a Node Outside the Cluster**.

Using the Spark Client

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, run the following command to authenticate the user. In normal mode, user authentication is not required.

kinit *Component service user*

Step 5 Run the Spark shell command. The following provides an example:

spark-beeline

----End

25.4 Accessing the Spark Web UI

The Spark web UI is used to view the running status of Spark applications. Google Chrome is recommended for better user experience.

Spark has two web UIs.

- Spark UI: used to display the status of running applications.

  The UI includes the following parts: Jobs, Stages, Storage, Environment, Executors, SQL, and JDBC/ODBC Server. The Streaming application has the Streaming tab in addition to the preceding parts.
- History Server UI: used to display the status of Spark applications that are complete or incomplete.

  The UI includes the application ID, application name, start time, end time, execution time, and owner information.

Spark UI

Step 1 Access the component management page.

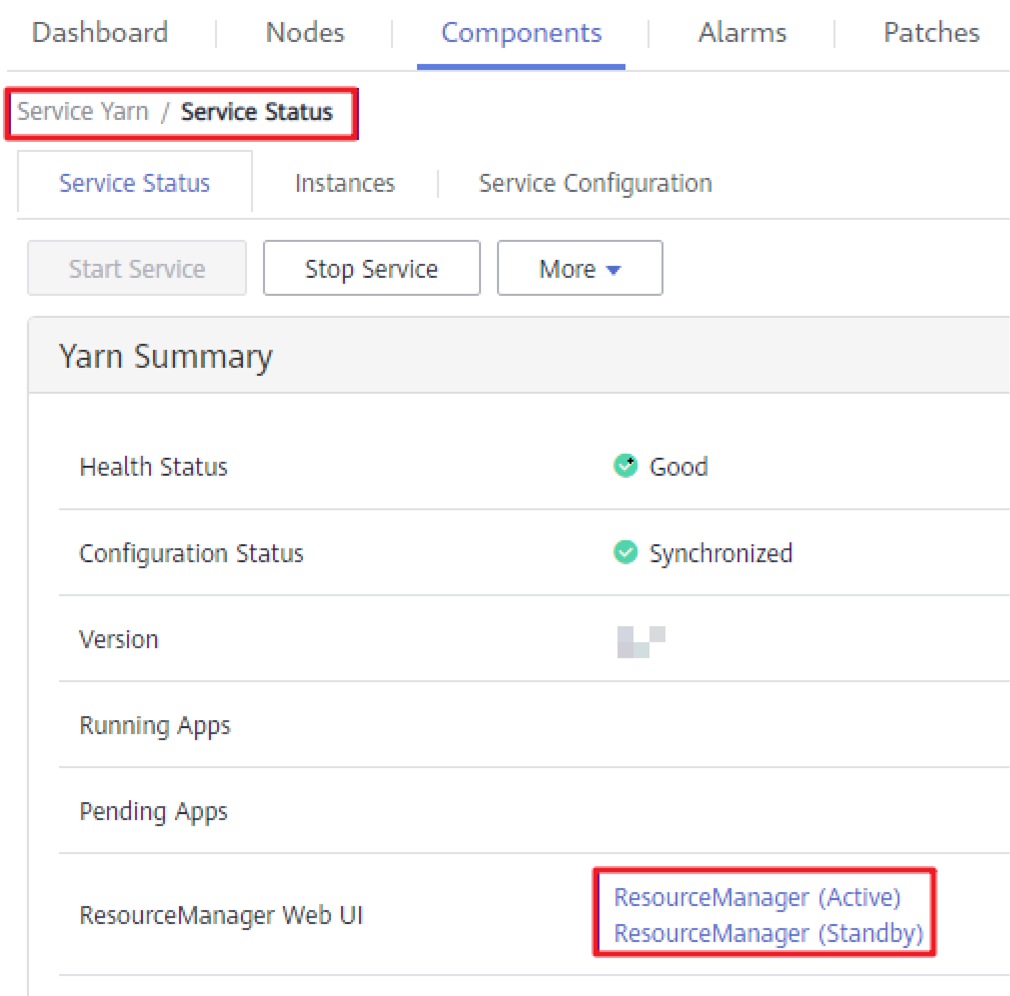
- For versions earlier than MRS 3.*x*, click the cluster name to go to the cluster details page and choose **Components**.

  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.)
- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > *Name of the desired cluster* > **Services**.

Step 2 Select **Yarn**. In the **Yarn Summary** area, click **ResourceManager** in **ResourceManager Web UI** to access the web UI.

Figure 25-9 ResourceManager web UI



Step 3 Locate the Spark application. Click **ApplicationMaster** in the last column of the application information. The Spark UI is displayed.

Figure 25-10 ApplicationMaster

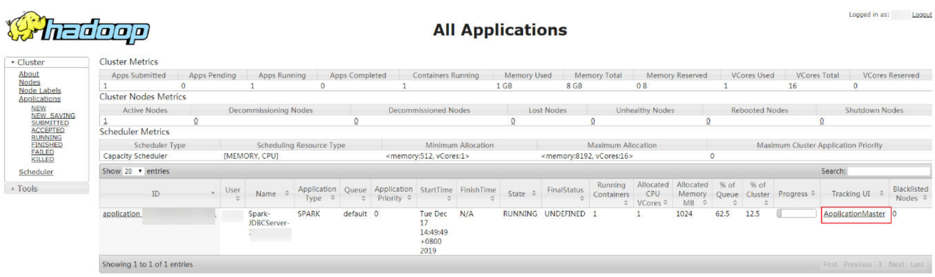


Figure 25-11 Spark UI

----End

History Server

Step 1 Access the component management page.

- For versions earlier than MRS 3.*x*, click the cluster name to go to the cluster details page and choose **Components**.

  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.)

- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > *Name of the desired cluster* > **Services**.

Step 2 Select **Spark**. In the **Spark Summary** area, click **JobHistory** corresponding to **Spark Web UI** to access the web UI.

Figure 25-12 ResourceManager web UI

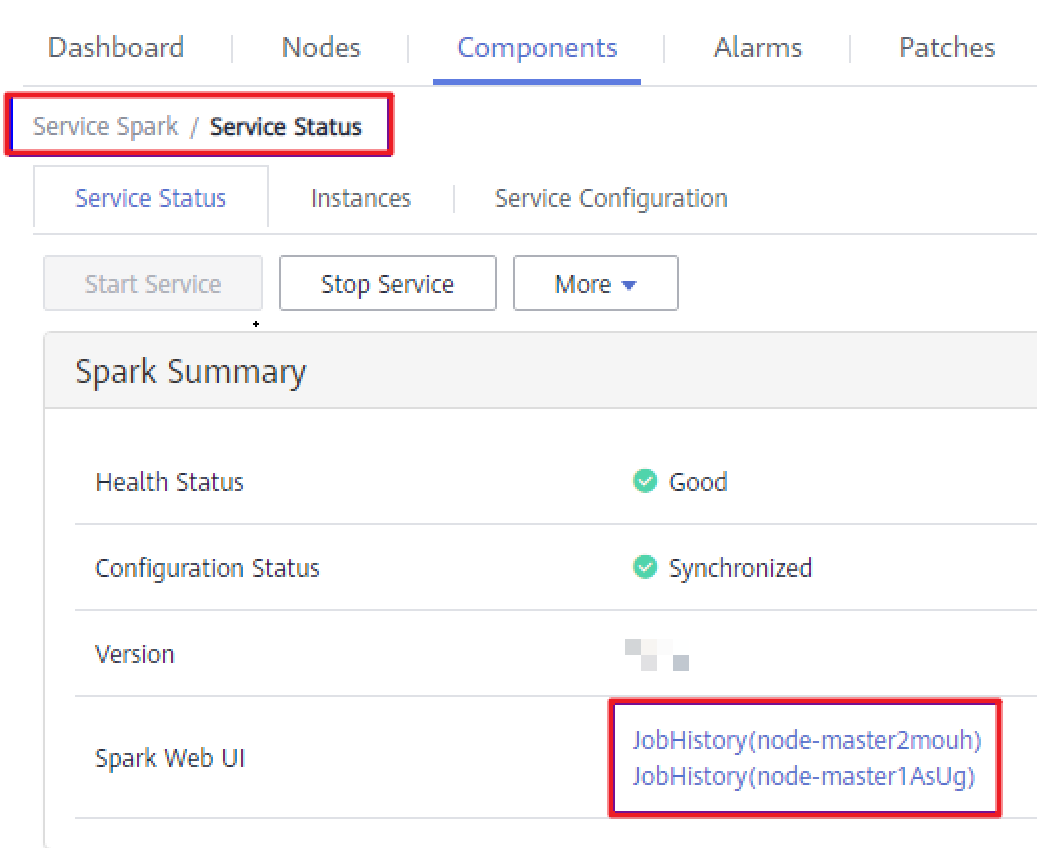
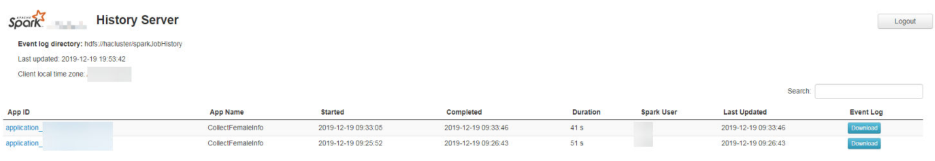


Figure 25-13 Spark History Server



----End

25.5 Interconnecting Spark with OpenTSDB

25.5.1 Creating a Table and Associating It with OpenTSDB

Function

MRS Spark can be used to access the data source of OpenTSDB, create and associate tables in the Spark, and query and insert the OpenTSDB data.

Use the **CREATE TABLE** command to create a table and associate it with an existing metric in OpenTSDB.

NOTE

If no metric exists in OpenTSDB, an error will be reported when the corresponding table is queried.

Syntax

```
CREATE TABLE [IF NOT EXISTS] OPENTSDB_TABLE_NAME   USING OPENTSDB OPTIONS (
'metric' = 'METRIC_NAME',
'tags' = 'TAG1,TAG2'
);
```

Keyword

Parameter	Description
metric	Indicates the name of the metric in OpenTSDB corresponding to the table to be created.
tags	Indicates the tags corresponding to the metric. The tags are used for classification, filtering, and quick retrieval. You can set 1 to 8 tags, which are separated by commas (,). The parameter value includes values of all tagKs in the corresponding metric.

Precautions

When creating a table, you do not need to specify the **timestamp** and **value** fields. The system automatically builds the following fields based on the specified tags. The fields **TAG1** and **TAG2** are specified by tags.

- TAG1 String
- TAG2 String
- timestamp Timestamp
- value double

Example

Create table **opentsdb_table** and associate it with metric **city.temp** of the OpenTSDB component.

```
CREATE table opentsdb_table using opentsdb OPTIONS ('metric'='city.temp',  'tags'='city,location');
```

25.5.2 Inserting Data to the OpenTSDB Table

Function

Run the **INSERT INTO** statement to insert the data in the table to the associated OpenTSDB metric.

Syntax

```
INSERT INTO TABLE_NAME SELECT * FROM SRC_TABLE;
INSERT INTO TABLE_NAME VALUES(XXX);
```

Keyword

Parameter	Description
TABLE_NAME	Indicates the name of the associated OpenTSDB table.
SRC_TABLE	Indicates the name of the table from which data is obtained. This parameter can be set to a name of a common table.

Precautions

- The inserted data cannot be **null**. If the inserted data is the same as the original data or only the **value** is different, the inserted data overwrites the original data.
- **INSERT OVERWRITE** is not supported.
- You are advised not to concurrently insert data into a table. If you concurrently insert data into a table, there is a possibility that conflicts occur, leading to data insertion failures.
- The **TIMESTAMP** format supports only yyyy-MM-dd hh:mm:ss.

Example

Insert data into table **opentsdb_table**.

```
insert into opentsdb_table values('city1','city2','2018-05-03 00:00:00',21);
```

25.5.3 Querying an OpenTSDB Table

This **SELECT** command is used to query data in an OpenTSDB table.

Syntax

```
SELECT * FROM table_name WHERE tagk=tagv LIMIT number;
```


Keyword

Parameter	Description
LIMIT	Used to limit the query results.
number	Only the INT type is supported.

Precautions

- The to-be-queried table must exist. Otherwise, an error is reported.
- The value of **tagv** must exist. Otherwise, an error occurs.

Example

Query data in the **opentsdb_table** table.

```
SELECT * FROM opentsdb_table LIMIT 100;
SELECT * FROM opentsdb_table WHERE city='city1';
```

25.5.4 Modifying the Default Configuration Data

By default, OpenTSDB connects to the local TSD process of the node where the Spark executor resides. In MRS, use the default configuration.

Table 25-1 OpenTSDB data source configuration

Parameter	Description	Example Value
spark.sql.datasource.opentsdb.host	Indicates the IP address of the connected TSD process.	Null (default value) **xx.xx.xx.xx** indicates the IP address. Separate multiple IP addresses with commas (,).
spark.sql.datasource.opentsdb.port	Indicates the port number of the TSD process.	4242 (default value)

Parameter	Description	Example Value
spark.sql.datasource.opentsdb.randomSeed	Indicates whether to use the random seed when the **spark.sql.datasource.opentsdb.host** is set to multiple addresses. If this parameter is set to **false**, all executors on the same node are connected to the same host. In this way, **spark.blacklist.enabled=true** can be used to implement task fault tolerance.	false (default value)

Example

Run the **set** statement in **spark-sql** and **spark-beeline**, and then run other SQL statements.

```
set spark.sql.datasource.opentsdb.host = 192.168.2.143,192.168.2.158;
SELECT * FROM opentsdb_table ;
```

26 Using Spark2x (for MRS 3.x or Later)

26.1 Spark User Permission Management

26.1.1 Spark SQL 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

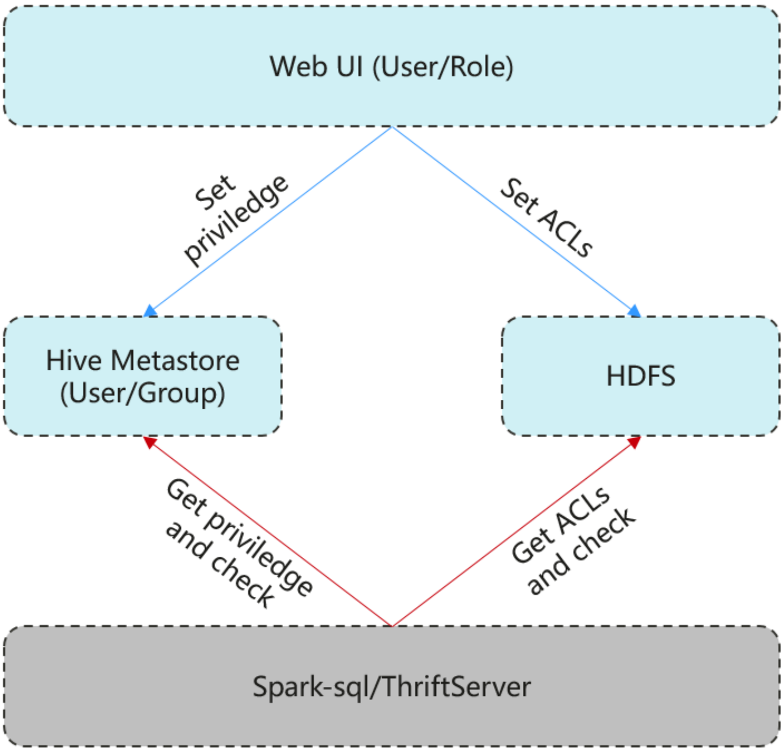
Spark SQL permission management indicates the permission system for managing and controlling users' operations on databases, to ensure that different users can operate databases separately and securely. A user can operate another user's tables and databases only with the corresponding permissions. Otherwise, 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 26-1 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 26-1 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 26-1 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 26-2 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.

26.1.2 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 26-3**.
   - **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 26-3 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**.

26.1.3 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.
- 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 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 26-4** describes the permission requirements.

Table 26-4 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** and **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	**Select** and **Insert**
SHOW COLUMNS	**Select**
SHOW TABLE PROPERTIES	**Select**
CREATE TABLE AS SELECT	**Select** and **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** and **Delete (for overwrite)**
LOAD	**Insert**, **Delete**, and 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	-

26.1.4 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 SparkSQL 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. 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. 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. Use a Spark SQL statement to query HBase table data.

select * from hbaseTable;

26.1.5 Configuring Spark2x 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, choose **Cluster** > ***Name of the desired cluster*** > **Services** > **Spark2x** > **Configurations**, click **All Configurations**, search for **acl**, and modify the following parameters on the JobHistory, JDBCServer, SparkResource, and Spark pages.

Table 26-5 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. All members in the list have the rights to manage all Spark tasks. You can configure multiple administrators and separate them from each other using commas (,).	admin

Parameter	Description	Default Value
spark.admin.acls.groups	Indicates the list of Spark administrator groups. All groups in the list have the permission to manage all Spark tasks. You can configure multiple administrator groups and separate them from each other using commas (,).	-
spark.modify.acls	Indicates the list of members that 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.

26.1.6 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 26-6 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 26-7 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 26-8 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.	-

26.2 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.huawei.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.

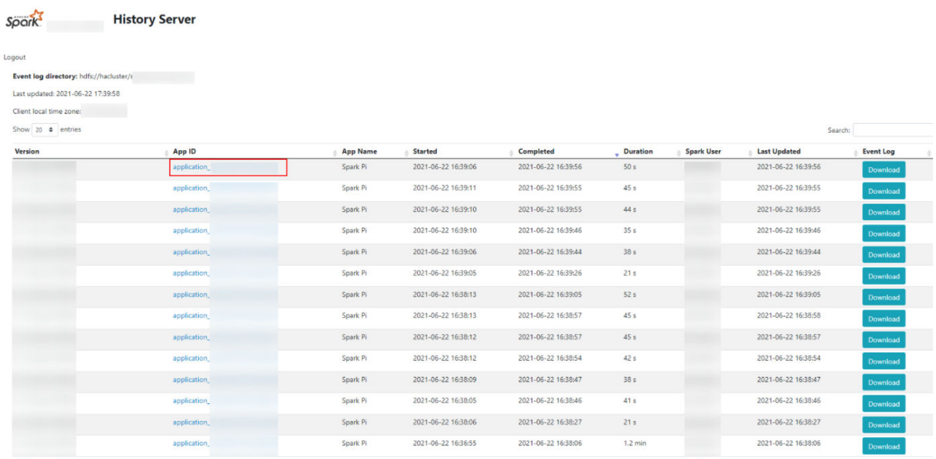
- Check the running status on the web page.

  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)** to go to the History Server page.

     The History Server UI is used to display the status of Spark applications that are complete or incomplete.

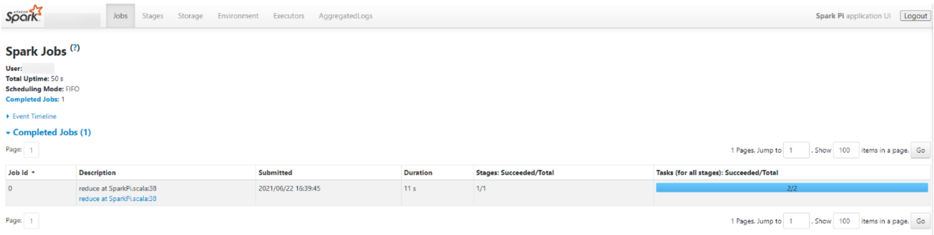
     Figure 26-2 History Server UI



  c. 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 26-3 Spark UI



- View Spark logs to learn application runtime conditions.

  You can view Spark2x logs to learn the application status and adjust the applications based on the logs. For details about Spark2x logs, see **Spark2x Logs**.

----End

26.3 Configuring Spark to Read HBase Data

Spark On HBase

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.

Step 1 Log in to Manager and choose **Cluster** > *Name of the desired 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** **>** *Name of the desired cluster* **>** **Service** **>** **Spark2x** **>** **Configuration** **>** **All Configurations** **>** **JDBCServer2x** **>** **Default**, and modify the following parameter.

Table 26-9 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 26-10 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. Run the following command to 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*

   Where **table1** indicates the name of the HBase table, and **/hperson** indicates the path where the CSV file is stored.

3. Run the following command to query data in spark-sql or spark-beeline, where *hbaseTable* is the corresponding Spark table name: The command is as follows:

   select * from *hbaseTable;*

----End

Spark on HBaseV2

Spark on HBase V2 allows users to query HBase tables in Spark SQL and to store data for HBase tables using the Beeline tool. You can use HBase APIs to create, read data from, and insert data into tables.

Step 1 Log in to Manager and choose **Cluster** > *Name of the desired 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 >** *Name of the desired cluster* **> Service > Spark2x > Configuration > All Configurations > JDBCServer2x > Default**, and modify the following parameter.

Table 26-11 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 26-12 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. Run the following command to 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

26.4 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 26-13 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

26.5 Spark Core Enterprise-Class Enhancements

26.5.1 Configuring Spark HA to Enhance HA

26.5.1.1 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, choose **Cluster** > *Name of the desired cluster* > **Services** > **Spark2x** > **Configurations**, click **All Configurations**, and search for and modify the following parameters.

Table 26-14 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

Parameter	Description	Default Value
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

26.5.1.2 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.

NOTE

If YARN resources are insufficient, you are advised not to enable the multi-tenant mode.

Configuration Description

Log in to Manager, choose **Cluster** > *Name of the desired cluster* > **Services** > **Spark2x** > **Configurations**, click **All Configurations**, and search for and modify the following parameters.

Table 26-15 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.

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.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
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

Parameter	Description	Default Value
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
hive.spark.client.server.connect.timeout	Specifies the timeout duration for client connections in multi-tenant mode.	5min

26.5.1.3 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, choose **Cluster** > *Name of the desired cluster* > **Service** > **Spark2x** > **Configuration**, click **All Configurations**, and search for and modify the following parameters.

Table 26-16 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.

26.5.2 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** > ***Name of the desired cluster*** > **Service** > **Spark2x** > **Configuration** and click **All Configurations**. Enter a parameter name in the search box.

Table 26-17 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.
```

26.5.3 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 > *Name of the desired cluster* > Service** > **Spark2x** > **Configuration**. Click **All Configurations** and enter a parameter name in the search box.

Table 26-18 Parameter description

Parameter	Description	Default Value
spark.sql.parquet.compression.codec	Used to set the compression format of a non-partitioned Parquet table.	snappy

26.5.4 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 in a client, for example, **/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

26.5.5 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. For example, 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 an error occurs indicating that the file does not exist when the query of data in a partitioned table is performed on the session spark-beeline/spark-sql that is always in the connected status. You can run the **refresh table** *Table name* command as prompted to query the data again.
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 path is **/tmp/SmallFilesLog.log4j**. To customize the log storage path, you can configure **log4j.appender.logfile.File** in **/opt/client/Spark2x/spark/tool/log4j.properties**.

26.5.6 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 26-19 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

Parameter	Description	Default Value
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
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.

26.6 Spark SQL Enterprise-Class Enhancements

26.6.1 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** > *Name of the desired cluster* > **Services** > **Spark2x**, click the **Configurations** tab and then **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]

Parameter	Description	Default Value	Value Range
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**.

26.6.2 Filtering Partitions without Paths in Partitioned Tables

Scenario

When you perform the ***select*** query in Hive partitioned tables, the **FileNotFoundException** exception is displayed if a specified partition path does not exist in HDFS. To avoid the preceding exception, configure **spark.sql.hive.verifyPartitionPath** parameter to filter partitions without paths.

Procedure

Perform either of the following methods to filter partitions without paths:

- Configure the following parameter in the **spark-defaults.conf** file on Spark client.

Table 26-20 Parameter description

Parameter	Description	Default Value
spark.sql.hive.verifyPartitionPath	Whether to filter partitions without paths when reading Hive partitioned tables. **true**: enables the filtering **false**: disables the filtering	false

- When running the spark-submit command to submit an application, configure the **--conf** parameter to filter partitions without paths.

  For example:

```
spark-submit --class org.apache.spark.examples.SparkPi  --conf spark.sql.hive.verifyPartitionPath=true
$SPARK_HOME/lib/spark-examples_*.jar
```

26.6.3 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** > *Name of the desired cluster* > **Services** > **Spark2x** > **Configurations**, click **All Configurations**, and search for the following parameter.

Parameter	Description	Default Value	Value Range
spark.sql.sources.partitionOverwriteMode	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]

26.6.4 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 proper 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

26.6.5 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 >** ***Name of the desired cluster*** **> Service** > **Spark2x** > **Configuration** and click **All Configurations**. Enter a parameter name in the search box.

Table 26-21 Parameter description

Parameter	Description	Default Value
spark.sql.shuffle.partitions	Indicates the default number of blocks divided during shuffle.	200

26.7 Spark Streaming Enterprise-Class Enhancements

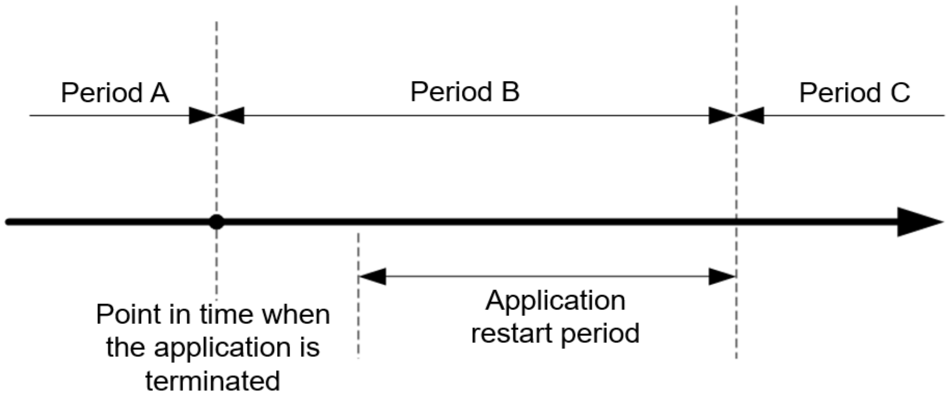
26.7.1 Configuring LIFO for Kafka

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 26-4 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 26-22 Parameter description

Parameter	Description	Default Value
spark.streaming.kafka.direct.lifo	Specifies whether to enable the LIFO function of Kafka.	false

Parameter	Description	Default Value
spark.streaming.kafka010.inputstream.class	Obtains the decoupled class on FusionInsight.	org.apache.spark.streaming.kafka010.HWDirectKafkaInputDStream

26.7.2 Configuring Reliability for Connected 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:

- If the offset of a topic in Kafka decreases when the Spark Streaming application is running, the latest offset of the topic in Kafka is used as the start position for reading Kafka data and subsequent data is read.

  For a task that has been generated but has not been scheduled, if the read Kafka offset is greater than the latest offset of the topic in Kafka, the task fails to be executed.

  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 26-23 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

26.8 Spark Core Performance Tuning

26.8.1 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")
```

26.8.2 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 ratio of the young generation and old 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)
  ```

26.8.3 Spark Core Memory Tuning

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
  ```

26.8.4 Configuring Spark Core Broadcasting Variables

Scenario

Broadcast distributes data sets to each node. It allows data to be obtained locally when a dataset is needed during a Spark task. If broadcast is not used, data serialization will be scheduled to tasks each time when a task requires data sets. It is time-consuming and makes the task get bigger.

1. If a dataset will be used by each slice of a task, 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
  ...
}
```

26.8.5 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 26-24 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

26.8.6 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** > *Name of the desired cluster* > **Services** > **Spark2x** > **Configurations**. Select **All Configurations**.

Step 3 Choose **SparkResource2x** > **Default** and modify the following parameters.

Table 26-25 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

26.8.7 Configuring Spark Dynamic Resource Scheduling in YARN Mode

Scenario

Resources are a key factor that affects Spark execution efficiency. When a long-running service (such as the JDBCServer) is allocated with multiple executors without tasks but resources of other applications are insufficient, resources are wasted and scheduled improperly.

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, and choose **Cluster** > *Name of the desired cluster* > **Service** > **Spark2x** > **Configuration** > **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 26-26 lists some optional configuration items.

Table 26-26 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.schedulerBacklogTimeout	Indicates the first timeout period for scheduling.	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.	60s
spark.dynamicAllocation.cachedExecutorIdleTimeout	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.

26.8.8 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.

26.8.9 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 26-5** shows the logic of implementation 1 and **Figure 26-6** shows logic of implementation 2.

Figure 26-5 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.

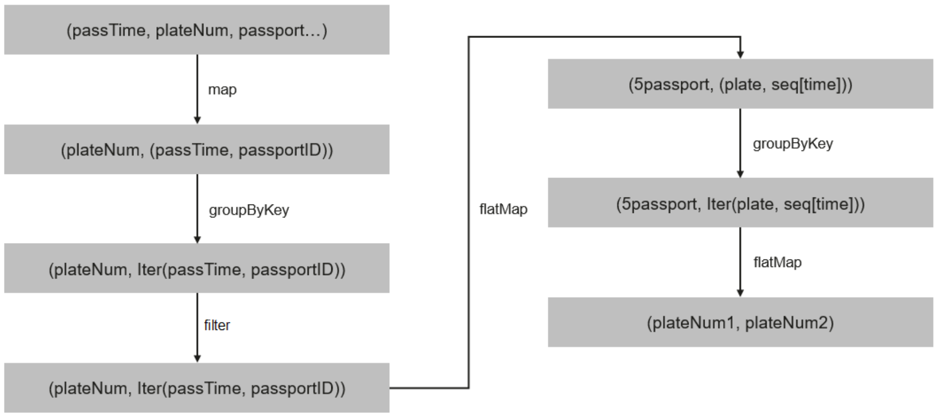
   (vehicle license plate number 1, vehicle license plate number 2), [(1st toll station, 5th toll station), (2nd toll station, 6th toll station), (1st toll station, 7th toll station), (3rd toll station, 8th toll station)]

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 26-6 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.

26.8.10 Experience

Use mapPartitions to calculate data by partition.

If the overhead of each record is high, for example:

```
rdd.map{x=>conn=getDBConn;conn.write(x.toString);conn.close}
```

Use mapPartitions to calculate data by partition.

```
rdd.mapPartitions(records => conn.getDBConn;for(item <- records)
write(item.toString); conn.close)
```

Use mapPartitions to flexibly operate data. For example, to calculate the TopN of a large data, mapPartitions can be used to calculate the TopN of each partition and then sort the TopN of all partitions when N is small. Compared with sorting full data for the TopN, this method has the higher efficiency.

Use 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 larger 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 smaller than the number of sections of the parent RDD.
- Use coalesce when the number of slices entered is too big to execute.
- Use coalesce when the programs are suspended in the shuffle operation because of a large number of tasks or the Linux resources are limited. In this case, use coalesce(numPartitions, true) to recreate partitions.

Configure a localDir for each disk.

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 a node is mounted with multiple disks, configure a Spark local Dir for each disk. This can effectively distribute shuffle files in multiple locations, improving disk I/O efficiency. The performance cannot be improved effectively if a disk is configured with multiple directories.

Collect small data sets.

The collect operation does not apply to a large data volume.

When the collect operation is performed, the Executor data will be sent to the Driver. Before performing this operation, ensure that the memory of Driver is sufficient. Otherwise, the Driver process may encounter an OutOfMemory error. If the data volume is unknown, perform the saveAsTextFile operation to write data into the HDFS. If the data volume is known and the Driver has sufficient memory, perform the collect operation.

Use reduceByKey

reduceByKey causes local aggregation on the Map side, which offers a smooth shuffle procedure. The shuffle operations, like groupByKey, will not perform aggregation on the Map side. Therefore, use reduceByKey as possible as you can, and avoid groupByKey().map(x=>(x._1,x._2.size)).

Broadcast map instead of array.

If table query is required for each record of the data transmitted from the Driver side, broadcast the data in the set/map instead of Iterator. The query speed of Set/Map is approximately O(1), while the query speed of Iterator is O(n).

Avoid data skew.

If data skew occurs (certain data volume is extremely large), the execution time of tasks is inconsistent even if there is no Garbage Collection (GC).

- Redefine the keys. Use keys of smaller granularity to optimize the task size.
- Modify the degree of parallelism (DOP).

Optimize the data structure.

- Store data by column. As a result, only the required columns are scanned when data is read.
- When using the Hash Shuffle, set **spark.shuffle.consolidateFiles** to **true** to combine the 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.

26.9 Spark SQL Performance Tuning

26.9.1 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 26-27 lists configuration parameters of the threshold for automatic broadcasting.

   Table 26-27 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.
     - If the values of the fields in a table are smaller than the threshold, the corresponding data in the table is broadcast.
     - If the values of the fields in the two tables are greater than the threshold, do not broadcast either of the table.
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.

26.9.2 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 with a large amount of data run slowly, resulting low computing performance. 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 and multiple tasks process data in one bucket, improving CPU usage and system performance.

NOTE

Data that is not skewed will be bucketed and run in its 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 26-28 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

26.9.3 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 26-29 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

26.9.4 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.

26.9.5 Optimizing Memory when Data Is Inserted into Dynamic Partitioned Tables

Scenario

When SparkSQL inserts data to 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, Garbage Collection (GC) is severe and Out of Memory (OOM) may occur.

Assume there are 10240 tasks and 2000 partitioned. Before the rename operation of HDFS files from a temporary directory to the target directory, there is about 29 GB FileStatus metadata.

Procedure

Insert **distribute by** followed by partition fields into dynamic partition statements.

For 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_retur n_tax,sr_return_amt_inc_tax,sr_fee,sr_return_ship_cost,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**;

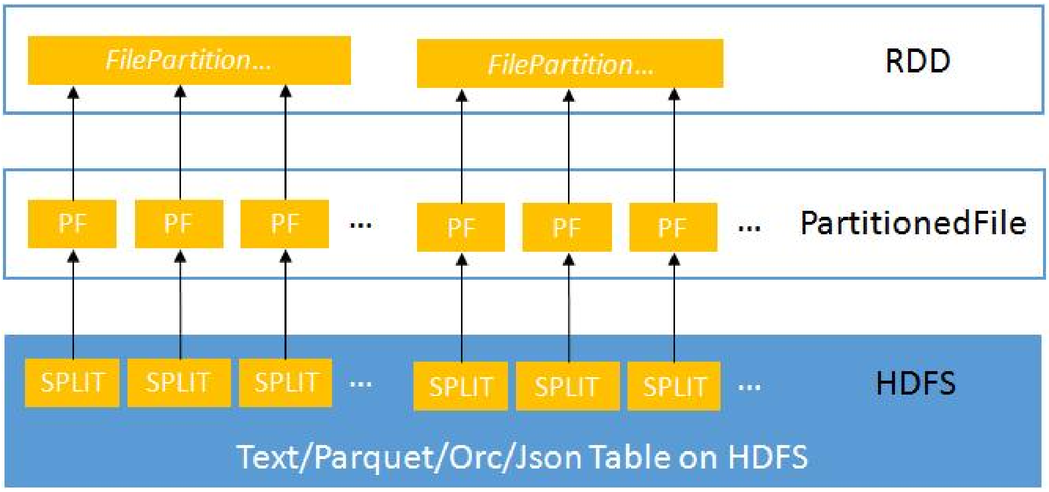
26.9.6 Optimizing Small Files

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 26-7**.

Figure 26-7 Merging small files



Procedure

If you want to enable small file optimization, configure the **spark-defaults.conf** file on the Spark client.

Table 26-30 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)

Parameter	Description	Default Value
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

26.9.7 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 26-31 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

26.9.8 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 26-32 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

26.9.9 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 26-33** in the **spark-defaults.conf** file on the Spark client.

Table 26-33 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
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

26.9.10 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 and choose **Cluster** > *Name of the desired cluster* > **Services** > **Spark2x**. On the page that is displayed, click the **Configuration** tab.
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)
```

26.10 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.

Streaming optimization is to optimize the performance of the three components.

- **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.

26.11 Spark O&M Management

26.11.1 Configuring 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 26-34 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
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

Configuration Item	Description	Default Value
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
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 26-35 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/8.1.0.1/spark-archive-2x.zip **NOTE** The version 8.1.0.1 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

Configuration Item	Description	Default Value or Configuration Example
spark.ssl.historyServer.enabled	Sets whether the history server uses SSL.	true
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.	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_8.1.0.1/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_8.1.0.1/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

26.11.2 Common 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 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 26-36 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 26-37 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 26-38 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 26-39 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 26-40 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 26-41 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 26-42 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 26-43 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.	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 26-44 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 26-45 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 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 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 26-46 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 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 26-47 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 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 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**.	-
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.	-

Parameter	Description	Default Value
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 26-48 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
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 26-49 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
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 26-50 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 26-51 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
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 26-52 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 26-53 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 (,).	-
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

Parameter	Description	Default Value
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 26-54 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 26-55 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 26-56 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 26-57 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 26-58 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 26-59 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.	-
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 26-60 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
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 authentication mode, the two communication parties share the same secret through a simple handshake.

Configure the following parameters in the **spark-defaults.conf** file on the Spark client.

Table 26-61 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 26-62 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

Parameter	Description	Default Value
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
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 26-63 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

26.11.3 Spark2x Logs

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 26-64 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

Log Type	Name	Description
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
	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 26-65 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 26-65 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** > *Name of the desired cluster* > **Service** > **Spark2x** > **Configuration**.

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 26-66 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)

26.11.4 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 26-67 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 26-68 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 26-68 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 26-69 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 26-69 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_8.1.0.1/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.

26.11.5 Viewing Container Logs on the Web UI

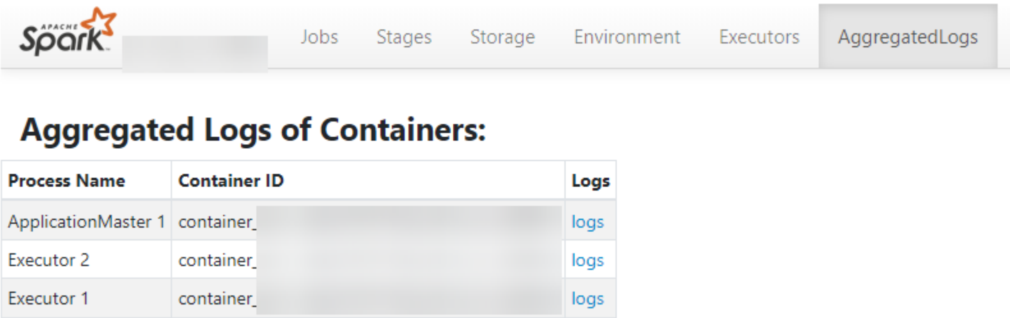
Scenarios

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 26-8**, the **AggregatedLogs** tab is added to the HistoryServer page. You can click **logs** to view aggregated logs.

Figure 26-8 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 26-70 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.	-

26.11.6 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

26.11.7 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

26.11.8 Configuring the Number of Lost Executors Displayed in WebUI

Scenario

In Spark WebUI, the **Executor** page can display information about Lost Executor. Executors are dynamically recycled. If the JDBCServer tasks are large, there may be too many lost executors displayed in WebUI. Therefore, the number of displayed lost executors can be configured.

Procedure

Configure the following parameter in the **spark-defaults.conf** file on Spark client.

Table 26-71 Parameter description

Parameter	Description	Default Value
spark.ui.retainedDeadExecutors	The maximum number of Lost Executors displayed in Spark WebUI.	100

26.11.9 Configuring Local Disk Cache for JobHistory

Scenarios

JobHistory can use local disks to cache the historical data of Spark applications to prevent the JobHistory memory from loading a large amount of application data, reducing the memory pressure. In addition, the cached data can be reused to improve the speed for subsequent application access.

Parameter Configuration

Log in to FusionInsight Manager, choose **Cluster** > *Name of the desired cluster* > **Services** > **Spark2x** > **Configurations**, click the **All Configurations** tab, and search for the following parameters:

Parameter	Description	Default Value
spark.history.store.path	Specifies the local directory for storing historical information for JobHistory. If this parameter is specified, JobHistory caches historical application data in the local disk instead of the memory.	${BIGDATA_HOME}/tmp/spark2x_JobHistory
spark.history.store.maxDiskUsage	Specifies the maximum available space of the local disk cache.	10 GB

26.11.10 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 26-72 Parameter description

Parameter	Scenario	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**.

26.11.11 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 26-73**.

- 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 26-73**, 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** > *Name of the desired cluster* > **Services** > **Spark2x** > **Configurations**. Click **All Configurations** and enter a parameter name in the search box.

Table 26-73 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}

26.11.12 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 and choose **Cluster** > *Name of the desired cluster* > **Services** > **Spark2x**. On the page that is displayed, click the

Configurations tab then the **All Configurations** sub-tab, 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]

26.11.13 Configuring the Column Statistics Histogram to Enhance the CBO Accuracy

Scenarios

The execution plan for SQL statements is optimized in Spark. Common optimization rules are heuristic optimization rules. Heuristic optimization rules are provided based on the characteristics of logical plans, and the data characteristics and the execution costs of operators are not considered. Spark 2.20 introduces the Cost-Based Optimization (CBO). CBO collects statistics on tables and columns and estimates the number of output records and size of each operator in bytes based on the input data sets of operators, which is the cost of executing an operator.

CBO will adjust the execution plan to minimize the end-to-end query time. The main points are as follows:

- Filter out irrelevant data as soon 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 sets of a specific operator based on the input data sets.

Table-level statistics includes: number of records and the total size of a table data file.

Column-level statistics includes: number of unique values, maximum value, minimum value, number of null values, average length, maximum length, and the histogram.

After the statistics is obtained, the execution cost of operators can be estimated. Common operators include 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). After the data is distributed, 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**.

Parameter Configuration

Log in to FusionInsight Manager, choose **Cluster >** *Name of the desired cluster* **> Services** > **Spark2x** > **Configurations**, click **All Configurations**, and search for the following parameters.

Parameter	Description	Default Value	Value Range
spark.sql.cbo.enabled	Enables CBO to estimate the statistics for the execution plan.	false	[true,false]
spark.sql.cbo.joinReorder.enabled	Enables CBO join for reordering.	false	[true,false]
spark.sql.cbo.joinReorder.dp.threshold	Specifies the maximum number of nodes that can be joined in the dynamic planning algorithm.	12	>=1
spark.sql.cbo.joinReorder.card.weight	Specifies the proportion of dimension (number of rows) in the comparison of planned cost during reconnection: Number of rows x Proportion of dimension + File size x (1 - Proportion of dimension)	0.7	0-1
spark.sql.statistics.size.autoUpdate.enabled	Enables the function of automatically updating the table size when the table data volume changes. Note: If there are a large number of data files in a table, this operation is time consume, and the data processing speed is reduced.	false	[true,false]
spark.sql.statistics.histogram.enabled	After this function is enabled, a histogram is generated when column statistics is collected. Histogram can improve the estimation accuracy, but collecting histogram information requires additional workload.	false	[true,false]

Parameter	Description	Default Value	Value Range
spark.sql.statistics.histogram.numBins	Specifies the number of bins for the generated histogram.	254	>=2
spark.sql.statistics.ndv.maxError	Specifies the maximum estimation deviation allowed by the HyperLogLog++ algorithm when the column level statistics is generated.	0.05	0-1
spark.sql.statistics.percentile.accuracy	Specifies the accuracy rate of the percentile estimation when the contour histogram is generated. The larger the value is, the more accurate the estimation is. The estimation error value can be obtained through (1.0/ Accuracy rate of the percentile estimation).	10000	>=1

NOTE

- If you want the histogram to take effect in CBO, the following conditions must be met:
  - Set **spark.sql.statistics.histogram.enabled** to **true**. The default value is **false**. Change the value to **true** to enable the histogram function.
  - Set **spark.sql.cbo.enabled** to **true**. The default value is **false**. Change the value to **true** to enable CBO.
  - Set **spark.sql.cbo.joinReorder.enabled** to **true**. The default value is **false**. Change the value to **true** to enable connection reordering.
- If a client is used to submit a task, you need to download the client again after configuring the following parameters: **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**.

26.11.14 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. For example, the installation directory is **/opt/client**. Go to the **/opt/client/Spark2x/spark/bin** directory and run **start-prequery.sh**.

Configure **prequeryParams.properties** by referring to **Table 26-74**.

Table 26-74 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
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.

26.12 Common Issues About Spark2x

26.12.1 Spark Core

26.12.1.1 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

For details, see **Viewing Container Logs on the Web UI**.

26.12.1.2 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:

http://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-common/yarn-default.xml

26.12.1.3 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.

26.12.1.4 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.

26.12.1.5 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.

26.12.1.6 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 26-75 describes the retry-related configuration items in the ResourceManager.

Table 26-75 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.

26.12.1.7 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.TransportResponseHandl er.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.j
ava:104)
        at
org.apache.spark.network.server.TransportChannelHandler.channelUnregistered(TransportChannelHandler.jav
a:94)
        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.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 26-76 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

26.12.1.8 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.

26.12.1.9 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.

26.12.1.10 "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. Spark closes the file handles for writing to partition files before a retry. If HDFS is unable to process the closure of numerous file handles promptly, the NameNode may not release the handles in time for the next retry, resulting in the "Failed to CREATE_FILE" exception.

However, this issue is typically transient and has minimal impact, as retries occur within milliseconds.

26.12.1.11 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.

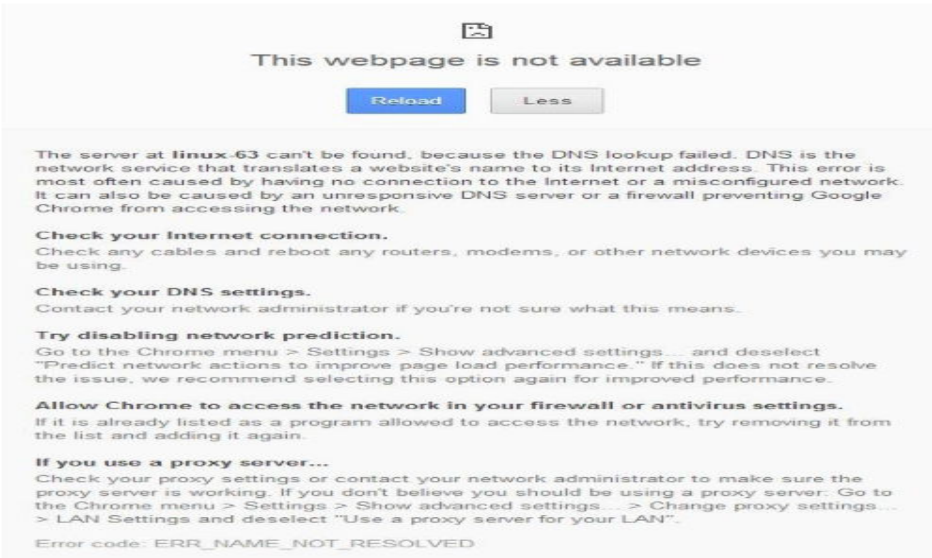
Sort shuffle, instead of Hash shuffle, is recommended to run a job.

26.12.1.12 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 26-9** is displayed.

Figure 26-9 DNS query failure



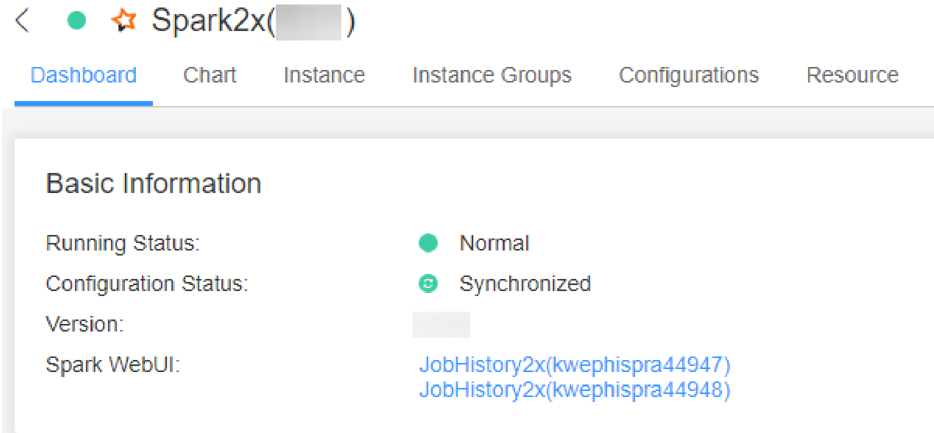
Answer

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 26-10**.

Figure 26-10 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.

26.12.1.13 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**.

26.12.1.14 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.

26.12.1.15 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 26-77 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

26.12.1.16 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:33
3)
        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.

Therefore, 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, click **Cluster**, click the name of the cluster, choose **Services** > **Yarn**, click **Configurations**, and click **All Configurations**. Click **NodeManager**, select **System**, and modify the configuration in the **GC_OPTS** parameter in the right pane.

Table 26-78 Parameters

Parameter	Description	Default Value
GC_OPTS	GC parameter of YARN NodeManager	128M

26.12.1.17 Why Does the Realm Information Fail to Be Obtained When SparkBench is Run on HiBench for the Cluster in Security Mode?

Question

Execution of the sparkbench task (for example, Wordcount) of HiBench6 fails. The bench.log indicates that the Yarn task fails to be executed. The failure information displayed on the Yarn UI is as follows:

```
Exception in thread "main" org.apache.spark.SparkException: Unable to load YARN support
  at org.apache.spark.deploy.SparkHadoopUtil$.liftedTree1$1(SparkHadoopUtil.scala:390)
```

```
 at org.apache.spark.deploy.SparkHadoopUtil$.yarn$lzycompute(SparkHadoopUtil.scala:385)
 at org.apache.spark.deploy.SparkHadoopUtil$.yarn(SparkHadoopUtil.scala:385)
 at org.apache.spark.deploy.SparkHadoopUtil$.get(SparkHadoopUtil.scala:410)
 at org.apache.spark.deploy.yarn.ApplicationMaster$.main(ApplicationMaster.scala:796)
 at org.apache.spark.deploy.yarn.ExecutorLauncher$.main(ApplicationMaster.scala:821)
 at org.apache.spark.deploy.yarn.ExecutorLauncher.main(ApplicationMaster.scala)
Caused by: java.lang.IllegalArgumentException: Can't get Kerberos realm
 at org.apache.hadoop.security.HadoopKerberosName.setConfiguration(HadoopKerberosName.java:65)
 at org.apache.hadoop.security.UserGroupInformation.initialize(UserGroupInformation.java:288)
 at org.apache.hadoop.security.UserGroupInformation.setConfiguration(UserGroupInformation.java:336)
 at org.apache.spark.deploy.SparkHadoopUtil.<init>(SparkHadoopUtil.scala:51)
 at org.apache.spark.deploy.yarn.YarnSparkHadoopUtil.<init>(YarnSparkHadoopUtil.scala:49)
 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 java.lang.Class.newInstance(Class.java:442)
 at org.apache.spark.deploy.SparkHadoopUtil$.liftedTree1$1(SparkHadoopUtil.scala:387)
 ... 6 more
Caused by: java.lang.reflect.InvocationTargetException
 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.security.authentication.util.KerberosUtil.getDefaultRealm(KerberosUtil.java:88)
 at org.apache.hadoop.security.HadoopKerberosName.setConfiguration(HadoopKerberosName.java:63)
 ... 16 more
Caused by: KrbException: Cannot locate default realm
 at sun.security.krb5.Config.getDefaultRealm(Config.java:1029)
 ... 22 more
```

Answer

In C80SPC200 and later, the file stored in the **/etc/krb5.conf** directory is no longer replaced during cluster installation. Instead, the file is stored in the corresponding path on the client through parameter configurations, and HiBench does not reference the client configuration file. Solution: Use the file stored in the **/opt/client/KrbClient/kerberos/var/krb5kdc/krb5.conf** directory on the client to overwrite that in the **/etc/krb5.conf** directories of all nodes. Make a backup before the overwriting.

26.12.2 Spark SQL and DataFrame

26.12.2.1 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.

26.12.2.2 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>
```

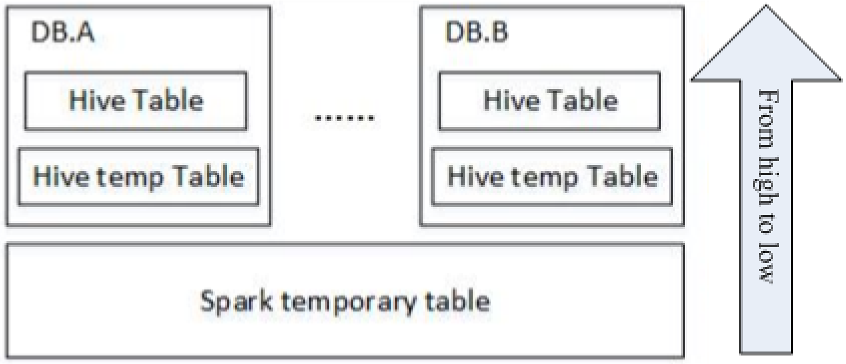
Answer

The table management hierarchy of Spark is shown in **Figure 26-11**. 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 26-11 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.

26.12.2.3 How to Assign a Parameter Value in a Spark Command?

Question

Is it possible to assign parameter values through Spark commands, in addition to through a user interface or a configuration file?

Answer

Spark configuration options can be defined either in a configuration file or in Spark commands.

To assign a parameter value, run the --conf command on a Spark client. The parameter value takes effect immediately after the command is run.

The command format is --conf + parameter name + parameter value. Example command:

--conf spark.eventQueue.size=50000

26.12.2.4 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.

26.12.2.5 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.

26.12.2.6 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.

26.12.2.7 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.

26.12.2.8 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 26-79 Parameter Description

Parameter	Description	Default Value
spark.sql.shuffle.partitions	Number of shuffle data blocks during the shuffle operation.	200

26.12.2.9 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 26-80 lists the default configuration.

Table 26-80 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 26-81 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

26.12.2.10 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. Do not name a table **table**.

26.12.2.11 Why Is a Task Suspended When the ANALYZE TABLE Statement Is Executed and Resources Are Insufficient?

Question

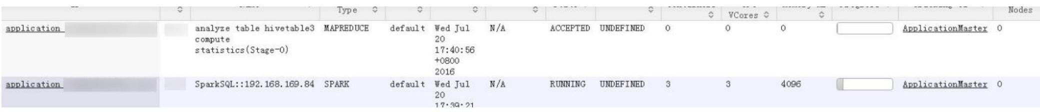
When the ***analyze table*** statement is executed using spark-sql, the task is suspended and the information below is displayed. Why?

```
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 26-12 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.

26.12.2.12 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**.

26.12.2.13 Why Do I Fail to Modify MetaData by Running the Hive Command?

Question

When do I fail to modify the metadata in the datasource and Spark on HBase table by running the Hive command?

Answer

The current Spark version does not support modifying the metadata in the datasource and Spark on HBase tables by running the Hive command.

26.12.2.14 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.

26.12.2.15 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 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.

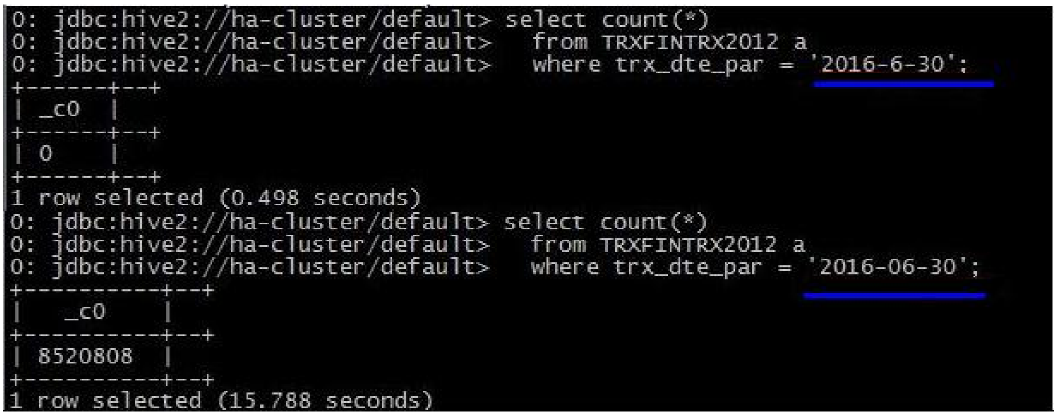
26.12.2.16 Why No Result Is found When 2016-6-30 Is Set in the Date Field as the Filter Condition?

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 26-13 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.

26.12.2.17 Why Does the --hivevar Option I Specified in the Command for Starting spark-beeline Fail to Take Effect?

Question

The **--hivevar** option specified in the command for starting spark-beeline fails to take effect.

If **--hivevar <VAR_NAME>=<var_value>** is used in the command for starting spark-beeline in an MRS cluster to customize a variable, no error is reported when spark-beeline is started. However, when the variable **<VAR_NAME>** is used in SQL statements, an error message is displayed because **<VAR_NAME>** cannot be parsed.

The following example describes the scenario:

1. Run the following command to start the spark-beeline:

   spark-beeline --hivevar <VAR_NAME>=<var_value>

2. Run the a SQL statement, for example, **DROP TABLE ${VAR_NAME}** in spark-beeline. The **VAR_NAME** exception is thrown.

Answer

The multi-session management function is added to the MRS cluster. Hive feature **--hivevar <VAR_NAME>=<var_value>** is no longer supported in Spark, and the **--hivevar** option in the spark-beeline startup command is invalid.

26.12.2.18 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.

26.12.2.19 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 26-48** in **Common 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 26-45** in **Common Parameters**.

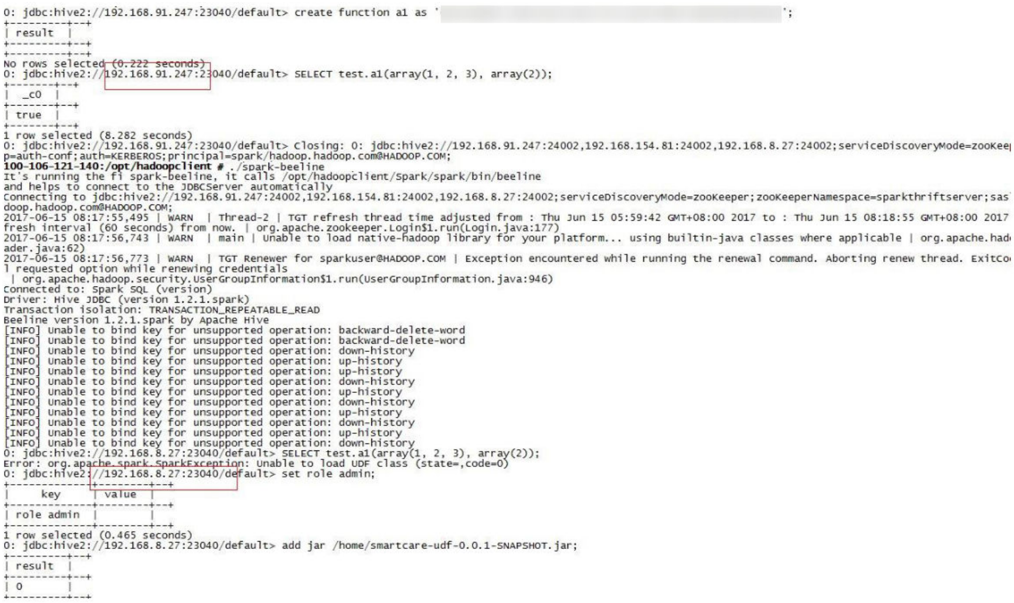
26.12.2.20 Why Functions Cannot Be Used When Different JDBCServers Are Connected?

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 26-14 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 26-15 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.

26.12.2.21 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.huawei.smartcare.dac.hive.udf.UDFArrayGreaterEqual';create function f12 as 'com.huawei.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 26-16 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;zooKe
eperNamespace=sparkthriftserver2x;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.hadoop.com@HA
DOOP.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.huawei.smartcare.dac.hive.udf.UDFArrayGreaterEqual';create function f12 as 'com.huawei.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 26-17 Output of the show functions command

  ```
  | datediff       |
  | day            |
  | dayofmonth     |
  | dayofyear      |
  | db2.f11        |
  | db2.f12        |
  | decimal        |
  | decode         |
  ```

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.

26.12.2.22 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.

26.12.2.23 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;

26.12.3 Spark Streaming

26.12.3.1 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.huawei.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.

26.12.3.2 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 26-18 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 26-19 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 26-18** 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.

26.12.3.3 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.

26.12.3.4 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 26-82**.

Table 26-82 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

26.12.3.5 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.

26.12.3.6 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 26-20**, **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 26-20 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

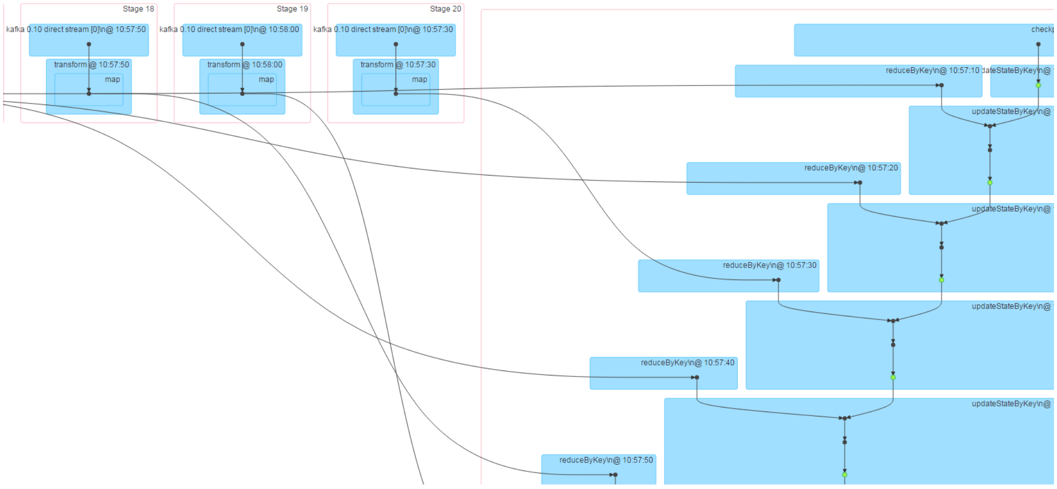
Answer

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 26-21**. 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 26-21**.

Figure 26-21 Mechanism for processing missing data during the restart



26.12.4 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 26-22**:

Figure 26-22 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.

26.12.5 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.

26.12.6 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 26-83 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

26.12.7 Why Is not an Application Displayed When I Run the Application with the Empty Part File?

Question

When I run an application with an empty part file in HDFS with the log grouping function enabled, why is not the application displayed on the homepage of JobHistory?

Answer

On the JobHistory page, information about applications is updated only with changed sizes of part files in HDFS. If a file is read for the first time, its size is compared with 0. The file is read only when the file size is greater than 0.

When the log grouping function is enabled, if the application you run does not have jobs in running status, the part file is empty. As a result, JobHistory does not read the part file and the application information is not displayed on the JobHistory page. However, if the size of part file is changed later, the application will be displayed on JobHistory.

26.12.8 Why Does Spark2x Fail to Export a Table with the Same Field Name?

Question

The following code fails to be executed on spark-shell of Spark2x:

```
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

In Spark2x, 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.

26.12.9 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**.

26.12.10 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.

26.12.11 How Does Spark2x Access External Cluster Components?

Question

There are two clusters, cluster 1 and cluster 2. How do I use Spark2x in cluster 1 to access 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 at the same time. Therefore, 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, the two HDFS services can be accessed at the same time.
- 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.

26.12.12 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.

26.12.13 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. After the application is complete, when I access the native page of the application in JobHistory, the browser waits for a long time before the page is displayed. If the native page cannot be displayed within 10 minutes, "Proxy Error" is displayed.

Figure 26-23 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.

26.12.14 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 26-24 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.        .hadoop.adapter.hdfs.plugin.HWAccessControlEnforce.checkPermission(                      .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:

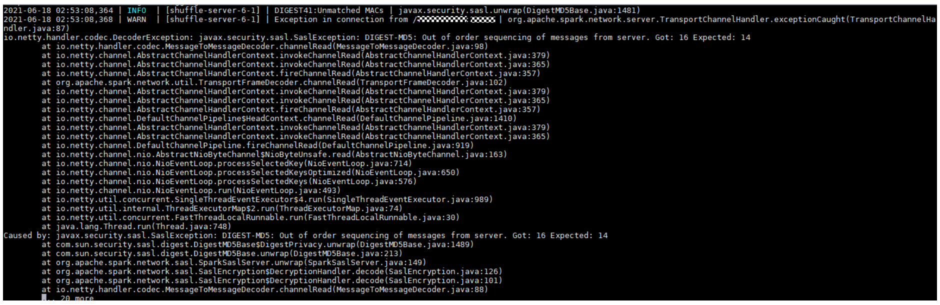
   vi ${BIGDATA_HOME}/FusionInsight_Spark2x_8.1.0.1/install/FusionInsight-Spark2x-*/spark/sbin/fake_prestart.sh

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.

26.12.15 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**.

27 Using Sqoop

27.1 Using Sqoop from Scratch

Sqoop is an open-source tool for transferring data between Hadoop (Hive) and traditional databases (such as MySQL and PostgreSQL). It can transfer data from a relational database (such as MySQL, Oracle, and PostgreSQL) to HDFS of Hadoop and the other way around.

Prerequisites

- You have selected the Sqoop component when creating a cluster of MRS 3.1.0 or later.
- You have installed the client. For details, see **Installing a Client (MRS 3.x or Later)**. For example, the installation directory of the client is **/opt/client**. The client directory in the following operations is an example. Change it to the actual installation directory.
- The corresponding driver package (for example, MySQL driver package **mysql-connector-java-5.1.47.jar**) exists in the client directory **/Sqoop/sqoop/lib**. If it does not exist, download it by referring to **3** in **Adapting Sqoop 1.4.7 to MRS 3.x Clusters**.

Exporting Data from HDFS to MySQL

Step 1 Log in to the node where the client is located.

Step 2 Run the following command to initialize environment variables:

source /opt/client/bigdata_env

Step 3 Run the following command to 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**

Table 27-1 Parameter description

Parameter	Description
--connect	Specifies the URL for connecting to JDBC. The value is in **jdbc:mysql://***IP address of the MySQL database*:*MySQL Port*/*Database name* format.
--username	Specifies the username for connecting to the MySQL database.
-password	Specifies the password for connecting to the MySQL database. 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.
-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.
-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.

Parameter	Description
-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 ensures the transaction security when data is imported to a relational database table. Due to multiple transactions during an import, this parameter can prevent other transactions from being affected when one transaction fails. For example, the imported data is incorrect or duplicate records exist.
-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 client is located.

Step 2 Run the following command to initialize environment variables:

source /opt/client/bigdata_env

Step 3 Run the following command to 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 27-2 Parameter description

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.
-append	Appends data to an existing dataset in the 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	Import data to a text file. After the text file is generated, you can run SQL statements in Hive to query the result.

Parameter	Description
-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.
-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**.

Parameter	Description
-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 last modified	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 (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**

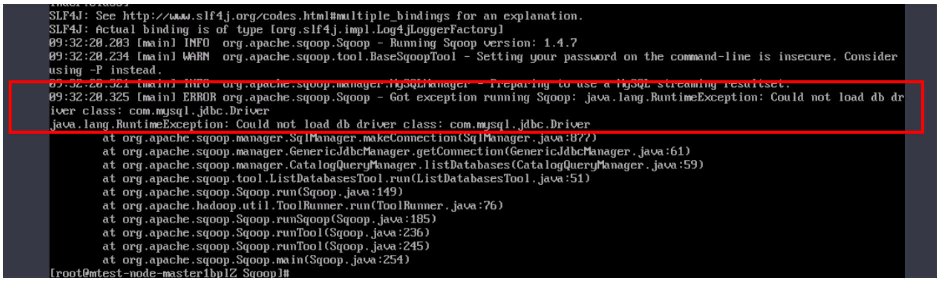
- Importing data from MySQL to OBS tables outside Hive

  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 27-1 An error indicating that the MySQL driver package is missing

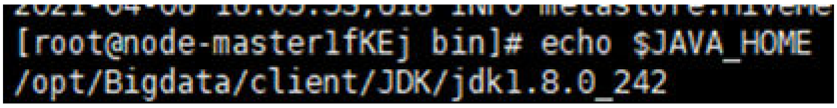


27.2 Adapting Sqoop 1.4.7 to MRS 3.*x* Clusters

Sqoop is a tool designed for efficiently transmitting a large amount of data between Apache Hadoop and structured databases (such as relational databases). Customers need to use Sqoop to migrate data in MRS. However, MRS of an earlier version does not provide Sqoop. This section describes how to install and use Sqoop. In MRS 3.1.0 or later, you can select the Sqoop component during cluster creation.

Prerequisites

The MRS client and the JDK environment have been installed.



Procedure

Step 1 [Download](#) the open-source **sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz** package.

Step 2 Save the downloaded package to the **/opt/Bigdata/client** directory on the node where the MRS client is installed and decompress it.

tar zxvf sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz

Step 3 Download the MySQL JDBC driver **mysql-connector-java-*xxx*.jar** from the MySQL official website. For details about how to select the MySQL JDBC driver, see the following table.

Table 27-3 Version information

JDBC Driver Version	MySQL Version
Connector/J 5.1	MySQL 4.1, MySQL 5.0, MySQL 5.1, and MySQL 6.0 alpha
Connector/J 5.0	MySQL 4.1, MySQL 5.0 servers, and distributed transaction (XA)
Connector/J 3.1	MySQL 4.1, MySQL 5.0 servers, and MySQL 5.0 except distributed transaction (XA)
Connector/J 3.0	MySQL 3.x and MySQL 4.1

Step 4 Put the MySQL driver package in the **/opt/Bigdata/client/sqoop-1.4.7.bin__hadoop-2.6.0/lib** directory of Sqoop and modify the owner group and permission of the JAR package. For details, see the owner group and permission of **omm:wheel** and **755** in **Figure 27-2**.

Figure 27-2 Owner group and permission of the MySQL driver package

```
-rwxr-xr-x. 1 omm wheel 1007502 Apr 28  2020 mysql-connector-java-5.1.47.jar
```

Step 5 Replace the JAR package in the **lib** directory of Sqoop with that starting with **jackson** in the **lib** directory of Hive on the MRS client, for example, **/opt/Bigdata/client/Hive/Beeline/lib**.

Figure 27-3 JAR package starting with **jackson**

```
-rwxr-xr-x. 1 omm wheel 1222059 Oct 19  2019 ivy-2.3.0.jar
-rwxr-xr-x. 1 omm wheel   46989 Apr 28  2020 jackson-annotations-2.6.3.jar
-rwxr-xr-x. 1 omm wheel  258876 Apr 28  2020 jackson-core-2.6.5.jar
-rwxr-xr-x. 1 omm wheel  232248 Apr 28  2020 jackson-core-asl-1.9.13.jar
-rwxr-xr-x. 1 omm wheel 1171380 Apr 28  2020 jackson-databind-2.6.5.jar
-rwxr-xr-x. 1 omm wheel   18336 Apr 28  2020 jackson-jaxrs-1.9.13.jar
-rwxr-xr-x. 1 omm wheel  780664 Apr 28  2020 jackson-mapper-asl-1.9.13.jar
-rwxr-xr-x. 1 omm wheel   27084 Apr 28  2020 jackson-xc-1.9.13.jar
```

Step 6 Copy the **jline** package from the **/opt/Bigdata/client/Hive/Beeline/lib** directory of the MRS Hive client to the **lib** directory of Sqoop.

Step 7 Run the **vim $JAVA_HOME/jre/lib/security/java.policy** command to add the following configuration:

permission javax.management.MBeanTrustPermission "register";

Step 8 Run the following commands to go to the **conf** directory of the Sqoop and add the configuration items of variables:

cd /opt/Bigdata/client/sqoop-1.4.7.bin__hadoop-2.6.0/conf

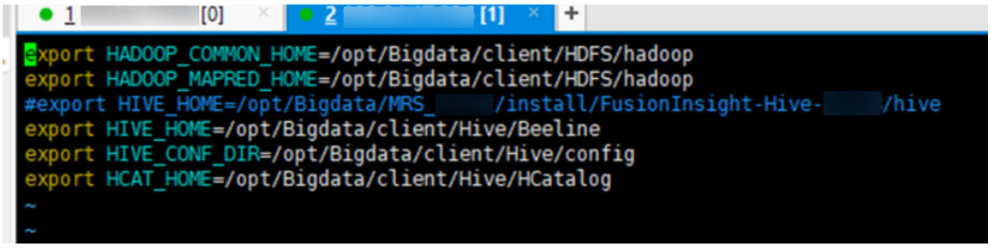
cp sqoop-env-template.sh sqoop-env.sh

Step 9 Run the **vim sqoop-env.sh** command to set the environment variables of Sqoop. Change the Hadoop and Hive directories as required.

```
export HADOOP_COMMON_HOME=/opt/Bigdata/client/HDFS/hadoop
export HADOOP_MAPRED_HOME=/opt/Bigdata/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/Bigdata/client/Hive/config
export HCAT_HOME=/opt/Bigdata/client/Hive/HCatalog
```

Figure 27-4 Setting environment variables of Sqoop



Step 10 Build the sqoop script. For example:

```
/opt/Bigdata/FusionInsight_Current/1_19_SqoopClient/install/FusionInsight-Sqoop-1.4.7/bin/sqoop import
--connect jdbc:mysql://192.168.0.183:3306/test
--driver com.mysql.jdbc.Driver
--username 'root'
--password 'xxx'
--query "SELECT id, name  FROM tbtest WHERE \$CONDITIONS"
--hcatalog-database default
--hcatalog-table test
--num-mappers 1
```

----End

27.3 Common Sqoop Commands and Parameters

Common Sqoop commands

Table 27-4 Common Sqoop 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.

Command	Description
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 27-5 Common parameters

Category	Parameter	Description
Parameters for database connection	--connect	Specifies the URL for connecting to a relational database.
	--connection-manager	Specifies the connection manager class.
	--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.

Category	Parameter	Description
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.
	--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.

Category	Parameter	Description
	--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.
	--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.

27.4 Common Issues About Sqoop

27.4.1 What Should I Do If Class QueryProvider Is Unavailable?

Question

What should I do if the QueryProvider class is unavailable?

```
2021-04-06 15:57:10,756 INFO manager.SqlManager: Using default fetchSize of 1000
2021-04-06 15:57:10,756 INFO tool.CodeGenTool: Beginning code generation
Apr 06, 2021 3:57:10 PM java.util.logging.LogManager$RootLogger log
SEVERE: Error loading factory org.apache.calcite.jdbc.CalciteJdbc41Factory
java.lang.NoClassDefFoundError: org/apache/calcite/linq4j/QueryProvider
        at java.lang.ClassLoader.defineClass1(Native Method)
        at java.lang.ClassLoader.defineClass(ClassLoader.java:757)
        at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
        at java.net.URLClassLoader.defineClass(URLClassLoader.java:468)
        at java.net.URLClassLoader.access$100(URLClassLoader.java:74)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:369)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:363)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:362)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:419)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:352)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:352)
        at java.lang.ClassLoader.defineClass1(Native Method)
        at java.lang.ClassLoader.defineClass(ClassLoader.java:757)
        at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
        at java.net.URLClassLoader.defineClass(URLClassLoader.java:468)
        at java.net.URLClassLoader.access$100(URLClassLoader.java:74)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:369)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:363)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:362)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:419)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:352)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:352)
        at java.lang.ClassLoader.defineClass1(Native Method)
        at java.lang.ClassLoader.defineClass(ClassLoader.java:757)
```

Answer

Search for the MRS client directory and save the following JAR packages to the **lib** directory of Sqoop.

```
-rwxr-xr-x. 1 omm wheel 4813045 Apr  6 15:56 calcite-core-1.19.0.jar
-rwxr-xr-x. 1 omm wheel  459944 Apr  6 16:01 calcite-linq4j-1.19.0.jar
```

27.4.2 What Should I Do If Method getHiveClient Does Not Exist?

Question

What should I do if the getHiveClient method does not exist when data is synchronized using HCatalog?

```
100000, related.hbase.service.name=HBase, hive.tez.min.partition.factor=0.1, hive.metastore.million.partition.optimizer.batch.delete.max=1000, hive-ext.dlcatalog.metastore.cacheable.client.cla
ss=org.apache.hadoop.hive.metastore.DLCatalogCacheableHiveMetaStoreClient}
Exception in thread "main" java.lang.NoSuchMethodError: org.apache.hive.hcatalog.common.HCatUtil.getHiveClient(Lorg/apache/hadoop/hive/conf/HiveConf;)Lorg/apache/hadoop/hive/metastore/HiveMeta
StoreClient;
        at org.apache.sqoop.mapreduce.hcat.SqoopHCatUtilities.isHCatView(SqoopHCatUtilities.java:178)
        at org.apache.sqoop.tool.BaseSqoopTool.validateHCatalogOptions(BaseSqoopTool.java:1655)
        at org.apache.sqoop.tool.ImportTool.validateOptions(ImportTool.java:1179)
        at org.apache.sqoop.Sqoop.run(Sqoop.java:137)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
        at org.apache.sqoop.Sqoop.runSqoop(Sqoop.java:183)
        at org.apache.sqoop.Sqoop.runTool(Sqoop.java:234)
        at org.apache.sqoop.Sqoop.runTool(Sqoop.java:243)
        at org.apache.sqoop.Sqoop.main(Sqoop.java:252)
[omm@node-master1fKEj bin]$ sh sqooptest.sh
```

Answer

Copy the JAR package from https://repo.huaweicloud.com/repository/maven/huaweicloudsdk/org/apache/hive/hcatalog/hive-hcatalog-core/ to the **HCatalog** directory of the MRS client and rename the original JAR package. As shown in the following figure, **302002.jar** is the new package and **310001-SNAPSHOT.jar.bak** is the original package with the suffix **.bak**.

Figure 27-5 HCatalog directory

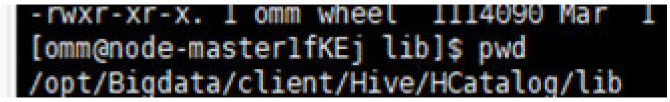
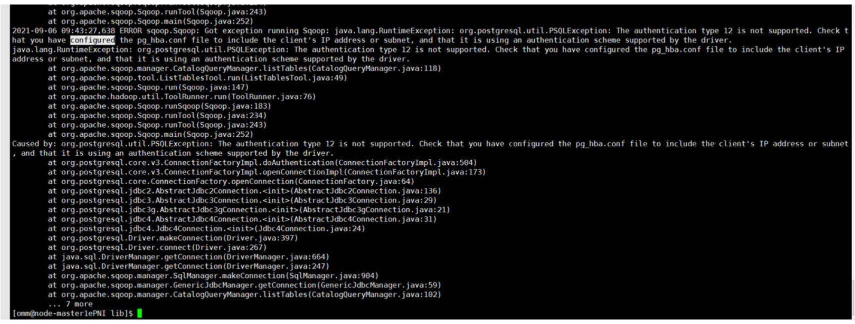


Figure 27-6 Replacing the JAR file

27.4.3 How Do I Do If PostgreSQL or GaussDB Fails to Connect?

Question

How do I do if PostgreSQL or GaussDB fails to connect?



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 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

    ■ 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:

    ■ 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 **/opt/Bigdata/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.
  - Solution:
        - 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 /opt/Bigdata/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 the error message "The authentication type 5 is not supported." is displayed when the **sqoop export** command is executed. 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 **/opt/Bigdata/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.
  - Solution:
    1. Move the driver package **gsjdbc4-*.jar** to the **tmp** directory on each node where MRS NodeManager instance is deployed.

       mv /opt/Bigdata/FusionInsight_HD_*/1_*_NodeManager/install/hadoop/share/hadoop/common/lib/gsjdbc4-*.jar /tmp

ii. Delete **/opt/Bigdata/client/Hive/Beeline/lib/gsjdbc4-*.jar**.

27.4.4 What Should I Do If Data Failed to Be Synchronized to a Hive Table on the OBS Using hive-table?

Question

What should I do if data failed to be synchronized to a Hive table on the OBS using hive-table?

```
2021-09-03 16:28:11,611 ERROR tools.DistCp: XAttrs not supported on at least one file system:
org.apache.hadoop.tools.CopyListing$XAttrsNotSupportedException: XAttrs not supported for file system:
obs://fdd-fs
        at org.apache.hadoop.tools.util.DistCpUtils.checkFileSystemXAttrSupport(DistCpUtils.java:555)
        at org.apache.hadoop.tools.DistCp.configureOutputFormat(DistCp.java:341)
        at org.apache.hadoop.tools.DistCp.createJob(DistCp.java:308)
        at org.apache.hadoop.tools.DistCp.createAndSubmitJob(DistCp.java:218)
        at org.apache.hadoop.tools.DistCp.execute(DistCp.java:197)
        at org.apache.hadoop.tools.DistCp.run(DistCp.java:155)
```

Answer

Change **-hive-table** to **-hcatalog-table**.

27.4.5 What Should I Do If Data Failed to Be Synchronized to an ORC or Parquet Table Using hive-table?

Question

What should I do if data failed to be synchronized to the ORC or parquet table using hive-table and error message that contains the **kite-sdk** package name is displayed?

Answer

Change **-hive-table** to **-hcatalog-table**.

27.4.6 What Should I Do If Data Failed to Be Synchronized Using hive-table?

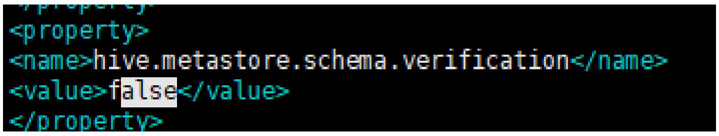
Question

What should I do if data failed to be synchronized using hive-table?

```
        ... 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.

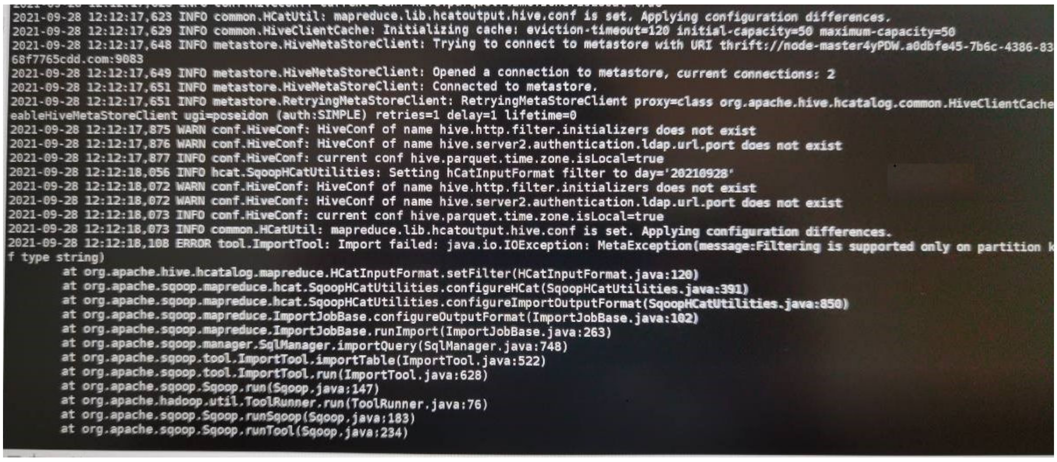


```
<property>
<name>hive.metastore.schema.verification</name>
<value>false</value>
</property>
```

27.4.7 What Should I Do If Data Failed to Be Synchronized to a Hive Parquet Table Using HCatalog?

Question

When the partition fields in a Hive parquet table are not of the string type, data in the table can be synchronized only using HCatalog. What should I do if the following error message is displayed during data synchronization?



```
2021-09-28 12:12:17,623 INFO common.HCatUtil: mapreduce.lib.hcatoutput.hive.conf is set. Applying configuration differences.
2021-09-28 12:12:17,629 INFO common.HiveClientCache: Initializing cache: eviction-timeout=120 initial-capacity=50 maximum-capacity=50
2021-09-28 12:12:17,648 INFO metastore.HiveMetaStoreClient: Trying to connect to metastore with URI thrift://node-master4yPDW.a0dbfe45-7b6c-4386-83
68f7765cdd.com:9083
2021-09-28 12:12:17,649 INFO metastore.HiveMetaStoreClient: Opened a connection to metastore, current connections: 2
2021-09-28 12:12:17,651 INFO metastore.HiveMetaStoreClient: Connected to metastore.
2021-09-28 12:12:17,651 INFO metastore.RetryingMetaStoreClient: RetryingMetaStoreClient proxy=class org.apache.hive.hcatalog.common.HiveClientCache
eableHiveMetaStoreClient ugi=poseidon (auth:SIMPLE) retries=1 delay=1 lifetime=0
2021-09-28 12:12:17,875 WARN conf.HiveConf: HiveConf of name hive.http.filter.initializers does not exist
2021-09-28 12:12:17,876 WARN conf.HiveConf: HiveConf of name hive.server2.authentication.ldap.url.port does not exist
2021-09-28 12:12:17,877 INFO conf.HiveConf: current conf hive.parquet.time.zone.isLocal=true
2021-09-28 12:12:18,056 INFO hcat.SqoopHCatUtilities: Setting hCatInputFormat filter to day='20210928'
2021-09-28 12:12:18,072 WARN conf.HiveConf: HiveConf of name hive.http.filter.initializers does not exist
2021-09-28 12:12:18,072 WARN conf.HiveConf: HiveConf of name hive.server2.authentication.ldap.url.port does not exist
2021-09-28 12:12:18,073 INFO conf.HiveConf: current conf hive.parquet.time.zone.isLocal=true
2021-09-28 12:12:18,073 INFO common.HCatUtil: mapreduce.lib.hcatoutput.hive.conf is set. Applying configuration differences.
2021-09-28 12:12:18,108 ERROR tool.ImportTool: Import failed: java.io.IOException: MetaException(message:Filtering is supported only on partition k
f type string)
        at org.apache.hive.hcatalog.mapreduce.HCatInputFormat.setFilter(HCatInputFormat.java:120)
        at org.apache.sqoop.mapreduce.hcat.SqoopHCatUtilities.configureHCat(SqoopHCatUtilities.java:391)
        at org.apache.sqoop.mapreduce.hcat.SqoopHCatUtilities.configureImportOutputFormat(SqoopHCatUtilities.java:850)
        at org.apache.sqoop.mapreduce.ImportJobBase.configureOutputFormat(ImportJobBase.java:102)
        at org.apache.sqoop.mapreduce.ImportJobBase.runImport(ImportJobBase.java:263)
        at org.apache.sqoop.manager.SqlManager.importQuery(SqlManager.java:748)
        at org.apache.sqoop.tool.ImportTool.importTable(ImportTool.java:522)
        at org.apache.sqoop.tool.ImportTool.run(ImportTool.java:628)
        at org.apache.sqoop.Sqoop.run(Sqoop.java:147)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
        at org.apache.sqoop.Sqoop.runSqoop(Sqoop.java:183)
        at org.apache.sqoop.Sqoop.runTool(Sqoop.java:234)
```

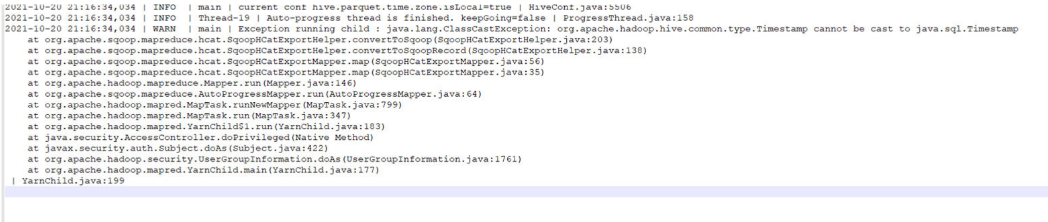
Answer

1. Delete the restricted code in the **SqoopHCatUtilities** class of Sqoop.
2. Change the value of the **hive.metastore.integral.jdo.pushdown** parameter in the **hive-site.xml** file on the Hive client to **true**.

27.4.8 What Should I Do If the Data Type of Fields timestamp and data Is Incorrect During Data Synchronization Between Hive and MySQL?

Question

What should I do if the data type of fields timestamp and data is incorrect during data synchronization between Hive and MySQL?



```
2021-10-20 21:16:34,034 | INFO  | main | current conf hive.parquet.time.zone.isLocal=true | HiveConf.java:5506
2021-10-20 21:16:34,034 | INFO  | Thread-19 | Auto-progress thread is finished. keepGoing=false | ProgressThread.java:158
2021-10-20 21:16:34,034 | WARN  | main | Exception running child : java.lang.ClassCastException: org.apache.hadoop.hive.common.type.Timestamp cannot be cast to java.sql.Timestamp
    at org.apache.sqoop.mapreduce.hcat.SqoopHCatExportHelper.convertToSqoop(SqoopHCatExportHelper.java:203)
    at org.apache.sqoop.mapreduce.hcat.SqoopHCatExportHelper.convertToSqoopRecord(SqoopHCatExportHelper.java:138)
    at org.apache.sqoop.mapreduce.hcat.SqoopHCatExportMapper.map(SqoopHCatExportMapper.java:56)
    at org.apache.sqoop.mapreduce.hcat.SqoopHCatExportMapper.map(SqoopHCatExportMapper.java:35)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:146)
    at org.apache.sqoop.mapreduce.AutoProgressMapper.run(AutoProgressMapper.java:64)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:799)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:347)
    at org.apache.hadoop.mapred.YarnChild$1.run(YarnChild.java:183)
    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.mapred.YarnChild.main(YarnChild.java:177)
 | YarnChild.java:199
```

Answer

- Forcibly convert the data type of the timestamp field in the Sqoop source package to be the same as that in Hive.
- Change the data type of the timestamp field in Hive to String.

28 Using Storm

28.1 Using Storm from Scratch

You can submit and delete Storm topologies on the MRS cluster client.

Prerequisites

The MRS cluster client has been installed, for example, in the **/opt/hadoopclient** directory. The client directory in the following operations is only an example. Change it based on the actual installation directory onsite.

Procedure

Step 1 Prepare the client based on service requirements. Log in to the node where the client is installed.

Log in to the node where the client is installed based on the client location. For details, see **Installing a Client**.

Step 2 Run the following command to switch to the client directory, for example, **/opt/hadoopclient**:

cd /opt/hadoopclient

Step 3 Run the following command to configure environment variables:

source bigdata_env

Step 4 For clusters with Kerberos authentication enabled, run the following command to authenticate the user. For clusters with Kerberos authentication disabled, skip this step.

kinit *Storm user*

Step 5 Run the following command to submit the Storm topology:

storm jar *Path of the topology package Class name of the topology Main method Topology name*

If the following information is displayed, the topology is submitted successfully.

```
Finished submitting topology: topo1
```

Step 6 Run the following command to query Storm topologies. For clusters with Kerberos authentication enabled, only users in the **stormadmin** or **storm** group can query all topologies.

storm list

Step 7 Run the following command to delete a Storm topology.

storm kill *Topology name*

----End

28.2 Using the Storm Client

Scenario

This section describes how to use the Storm client in an O&M scenario or service scenario.

Prerequisites

- You have installed the client. For example, the installation directory is **/opt/hadoopclient**.
- Service component users have been created by the MRS cluster administrator. In security mode, machine-machine users have downloaded the keytab file. A human-machine user must change the password upon the first login. (Not involved in normal mode)

Procedure

Step 1 Prepare the client based on service requirements. Log in to the node where the client is installed.

Log in to the node where the client is installed based on the client location. For details, see **Installing a Client**.

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 If multiple Storm instances are installed, run the following command to load the environment variables of a specific instance when running the Storm command to submit the topology. Otherwise, skip this step. The following command uses the instance Storm-2 as an example.

source Storm-2/component_env

Step 5 Run the following command to perform user authentication (skip this step in normal mode):

kinit *Component service user*

Step 6 Run the following command to perform operations on the client:

For example, run the following command:

- **cql**
- **storm**

NOTE

A Storm client cannot be connected to secure and non-secure ZooKeepers at the same time.

----End

28.3 Submitting Storm Topologies on the Client

Scenario

You can submit Storm topologies on the cluster client to continuously process stream data. For clusters with Kerberos authentication enabled, users who submit topologies must be members of the **stormadmin** or **storm** group.

Prerequisites

The client has been updated.

Procedure

Step 1 Prepare the client based on service requirements. Log in to the node where the client is installed.

Log in to the node where the client is installed based on the client location. For details, see **Installing a Client**.

Step 2 Run the following command to set the permissions on the topology JAR file:

For example, run the following command to change the permissions on **/opt/storm/topology.jar**:

chmod 600 /opt/storm/topology.jar

Step 3 Run the following command to switch to the client directory, for example, **/opt/client**.

cd /opt/client

Step 4 Run the following command to configure environment variables:

source bigdata_env

Step 5 If multiple Storm instances are installed, run the following command to load the environment variables of a specific instance when running the Storm command to submit the topology. Otherwise, skip this step. The following command uses the instance Storm-2 as an example.

source Storm-2/component_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 *Storm user*

Step 7 For versions earlier than MRS 3.x, run the following command to submit the Storm topology:

storm jar *Path of the topology package Class name of the topology Main method Topology name*

If the following information is displayed, the topology is submitted successfully.

```
Finished submitting topology: topo1
```

NOTE

- To support sampling messages, add the **topology.debug** and **topology.eventlogger.executors** parameters.
- Data processing methods vary with topologies. The topology in the example generates characters randomly and separates character strings. To query the processing status, enable the sampling function and perform operations according to **Querying Storm Topology Logs**.

Step 8 Run the following command to submit a topology task for MRS 3.x or later:

storm jar *topology-jar-path class input parameter list*

- *topology-jar-path* indicates the path of the JAR file of the topology.
- *class* indicates the class name of the main method used by the topology.
- *Input parameter list* includes input parameters of the main method used by the topology.

For example, run the following command to submit the **/opt/storm/topology.jar** topology calculated by WordCount and use the topology name as the input parameter:

storm jar /opt/storm/topology.jar com.huawei.storm.example.WordCountTopology topology1

If the following information is displayed, the topology is submitted successfully:

```
Finished submitting topology: topology1
```

NOTE

- The login authentication user must correspond to the loaded environment variable (**component_env**). Otherwise, an error occurs when you run the **storm** command to submit the topology task.
- After the client environment variable is loaded and the corresponding user login succeeds, the user can run the Storm command on any Storm client to submit the topology task. After the command is executed, the successfully submitted topology is still in the Storm cluster of the user.
- If the cluster domain name is changed, you need to reset the domain name before submitting the topology. Run the cql statement, for example, **set "kerberos.domain.name" = "***hadoop.huawei.com***"**.

Step 9 Run the following command to query Storm topologies. For clusters with Kerberos authentication enabled, only users in the **stormadmin** or **storm** group can query all topologies.

storm list

----End

28.4 Accessing the Storm Web UI

Scenario

The Storm web UI provides a graphical interface for using Storm.

The following information can be queried on the Storm web UI:

- Storm cluster summary
- Nimbus summary
- Topology summary
- Supervisor summary
- Nimbus configurations

Prerequisites

- The password of user **admin** has been obtained. The password of user **admin** is specified by you during the cluster creation.
- If a user other than **admin** is used to access the Storm web UI, the user must be added to the **storm** or **stormadmin** user group.

Procedure

Step 1 Access the component management page.

- For versions earlier than MRS 3.*x*, click the cluster name to go to the cluster details page and choose **Components**.

  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.)

- For MRS 3.*x* or later, log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. Choose **Cluster** > *Name of the desired cluster* > **Services**.

Step 2 Log in to the Storm WebUI.

- For versions earlier than MRS 3.x: Choose **Storm**. On the **Storm Summary** area, click any UI link on the right side of **Storm Web UI** to open the Storm web UI.

  NOTE

  When accessing the Storm web UI for the first time, you must add the address to the trusted site list.

- For MRS 3.x or later, choose **Storm** > **Overview**. In the **Basic Information** area, click any UI link on the right side of **Storm Web UI** to open the Storm web UI.

----End

Related Tasks

- Click a topology name to view details, status, Spouts information, Bolts information, and configuration information of the topology.
- In the **Topology actions** area, click **Activate**, **Deactivate**, **Rebalance**, **Kill**, **Debug**, **Stop Debug**, and **Change Log Level** to activate, deactivate, redeploy, delete, debug, and stop debugging the topology, and modify the log levels, respectively. You need to set the waiting time for the redeployment and deletion operations. The unit is second.
- In the **Topology Visualization** area, click **Show Visualization** to visualize a topology. After the topology is visualized, the WebUI displays the topology structure.

28.5 Managing Storm Topologies

Scenario

You can manage Storm topologies on the Storm web UI. Users in the **storm** group can manage only the topology tasks submitted by themselves, while users in the **stormadmin** group can manage all topology tasks.

Procedure

Step 1 For details about how to access the Storm WebUI, see **Accessing the Storm Web UI**.

Step 2 In the **Topology summary** area, click the desired topology.

Step 3 Use options in **Topology actions** to manage the Storm topology.

- Activating a topology

  Click **Activate** to activate the topology.
- Deactivating a topology

  Click **Deactivate** to deactivate the topology.
- Re-deploying a topology

  Click **Rebalance** and specify the wait time (in seconds) of re-deployment. Generally, if the number of nodes in a cluster changes, the topology can be re-deployed to maximize resource usage.
- Deleting a topology

  Click **Kill** and specify the wait time (in seconds) of the deletion.
- Starting or stopping sampling messages

  Click **Debug**. In the dialog box displayed, specify the percentage of the sampled data volume. For example, if the value is set to **10**, 10% of data is sampled.

  To stop sampling, click **Stop Debug**.

  NOTE

  This function is available only if the sampling function is enabled when the topology is submitted. For details about querying data processing information, see **Querying Storm Topology Logs**.

- Modifying the topology log level

  Click **Change Log Level** to specify a new log level.

Step 4 Displaying a topology

In the **Topology Visualization** area, click **Show Visualization** to visualize the topology.

----End

28.6 Querying Storm Topology Logs

Scenario

You can query topology logs to check the execution of a Storm topology in a worker process. To query the data processing logs of a topology, enable the **Debug** function when submitting the topology. Only streaming clusters with Kerberos authentication enabled support this function. In addition, the user who queries topology logs must be the one who submits the topology or a member of the **stormadmin** group.

Prerequisites

- The network of the working environment has been configured.
- The sampling function has been enabled for the topology.

Querying Worker Process Logs

Step 1 For details about how to access the Storm WebUI, see **Accessing the Storm Web UI**.

Step 2 In the **Topology Summary** area, click the desired topology to view details.

Step 3 Click the desired **Spouts** or **Bolts** task. In the **Executors (All time)** area, click a port in **Port** to view detailed logs.

----End

Querying Data Processing Logs of a Topology

Step 1 For details about how to access the Storm WebUI, see **Accessing the Storm Web UI**.

Step 2 In the **Topology Summary** area, click the desired topology to view details.

Step 3 Click **Debug**, specify the data sampling ratio, and click **OK**.

Step 4 Click the **Spouts** or **Bolts** task. In **Component summary**, click **events** to view data processing logs.

----End

28.7 Storm Common Parameters

This section applies to MRS 3.*x* or later.

Navigation Path

For details about how to set parameters, see **Modifying Cluster Service Configuration Parameters**.

Parameter Description

Table 28-1 Parameter description

Parameter	Description	Default Value
supervisor.slots.ports	Specifies the list of ports that can run workers on the supervisor. Each worker occupies a port, and each port runs only one worker. This parameter is used to set the number of workers that can run on each server. Ports range from 1024 to 65535, and ports are separated by commas (,).	6700,6701,6702,6703
WORKER_GC_OPTS	Specifies the JVM option used for supervisor to start worker. It is recommended that you set this parameter based on memory usage of a service. For simple service processing, the recommended value is **-Xmx1G**. If window cache is used, the value of this parameter is calculated based on the following formula: Size of each record x Period x 2	-Xms1G -Xmx1G -XX:+UseG1GC -XX:+PrintGCDetails -Xloggc:artifacts/gc.log -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=artifacts/heapdump

Parameter	Description	Default Value
default.schedule.mode	Specifies the default scheduling mode of the scheduler. Options are as follows: ● **AVERAGE**: indicates that the scheduling mechanism that uses the number of idle slots as the priority is used. ● **RATE**: indicates that the scheduling mechanism that uses the rate of idle slots as the priority is used.	AVERAGE
nimbus.thrift.threads	Set the maximum number of connection threads when the active Nimbus externally provides services. If the Storm cluster is large and the number of Supervisor instances is large, increase connection threads.	512

28.8 Configuring a Storm Service User Password Policy

Scenario

This section applies to MRS 3.*x* or later.

After submitting a topology task, a Storm service user must ensure that the task continuously runs. During topology running, the worker process may need to restart to ensure continuous topology work. If the password of a service user is changed or the number of days that a password is used exceeds the maximum number specified in a password policy, topology running may be affected. An MRS cluster administrator must configure a separate password policy for Storm service users based on enterprise security requirements.

NOTE

If a separate password policy is not configured for Storm service users, an old topology can be deleted and then submitted again after a service user password is changed so that the topology can continuous run.

Impact on the System

- After a separate password policy is configured for a Storm service user, the user is not affected by **Password Policy** on the Manager page.
- If a separate password policy is configured for a Storm service user and cross-cluster entrusted relationships are configured, a password must be reset for the Storm service user on Manager based on the password policy.

Prerequisites

An MRS cluster administrator has understood service requirements and created a Human-Machine user, for example, **testpol**.

Procedure

Step 1 Log in to any node in the cluster as user **omm**.

Step 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.

Step 3 Run the following commands to export the environment variables:

EXECUTABLE_HOME="${CONTROLLER_HOME}/kerberos_user_specific_binay/kerberos"

LD_LIBRARY_PATH=${EXECUTABLE_HOME}/lib:$LD_LIBRARY_PATH

PATH=${EXECUTABLE_HOME}/bin:$PATH

Step 4 Run the following command and enter the Kerberos administrator password to log in to the Kerberos console:

kadmin -p kadmin/admin

NOTE

For initial use, the **kadmin/admin** password must be changed for the **kadmin/admin** user.

If the following information is displayed, you have successfully logged in to the Kerberos console.

```
kadmin:
```

Step 5 Run the following command to check details about the created **Human-Machine** user:

getprinc*Username*

Sample command for viewing details about the **testpol** user:

getprinc testpol

If the following information is displayed, the specified user has used the default password policy:

```
Principal: testpol@<System domain name>
......
Policy: default
```

Step 6 Run the following command to create a separate password policy, such as **streampol**, for the Storm service user:

addpol -maxlife 0day -minlife 0sec -history 1 -maxfailure 5 -failurecountinterval 5min -lockoutduration 5min -minlength 8 -minclasses 4 streampol

In the command, **-maxlife** indicates the maximum validity period of a password, and **0day** indicates that a password will never expire.

Step 7 Run the following command to view the newly created policy **streampol**:

getpol streampol

If the following information is displayed, the new policy specifies that the password will never expire:

```
Policy: streampol
 Maximum password life: 0 days 00:00:00
......
```

Step 8 Run the following command to apply the new policy **streampol** to the **testpol** Storm user:

modprinc -policy streampol testpol

In the command, **streampol** indicates a policy name, and **testpol** indicates a username.

If the following information is displayed, the properties of the specified user have been modified:

```
Principal "testpol@<System domain name>" modified.
```

Step 9 Run the following command to view current information about the **testpol** Storm user:

getprinc testpol

If the following information is displayed, the specified user has used the new password policy:

```
Principal: testpol@<System domain name>
......
 Policy: streampol
```

----End

28.9 Migrating Storm Services to Flink

28.9.1 Overview

This section applies to MRS 3.*x* or later.

From 0.10.0, Flink provides a set of APIs to smoothly migrate services compiled using Storm APIs to the Flink platform. This can be used in most of the service scenarios.

Flink supports the following service migration modes:

1. Complete migration of Storm services: Convert and run a complete Storm topology developed by Storm APIs.

2. Embedded migration of Storm services: Storm code is embedded in DataStream of Flink, for example, Spout/Bolt compiled using Storm APIs.

Flink provides the flink-storm package for the preceding service migration.

28.9.2 Completely Migrating Storm Services

Scenarios

This section describes how to convert and run a complete Storm topology developed using Storm API.

Procedure

Step 1 Open the Storm service project, modify the POM file of the project, and add the reference of **flink-storm_2.11**, **flink-core**, and **flink-streaming-java_2.11**. The following figure shows an example.

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-storm_2.11</artifactId>
    <version>1.4.0</version>
    <exclusions>
        <exclusion>
            <groupId>*</groupId>
            <artifactId>*</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-core</artifactId>
    <version>1.4.0</version>
    <exclusions>
        <exclusion>
            <groupId>*</groupId>
            <artifactId>*</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-streaming-java_2.11</artifactId>
    <version>1.4.0</version>
    <exclusions>
        <exclusion>
            <groupId>*</groupId>
            <artifactId>*</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

NOTE

If the project is not a non-Maven project, manually collect the preceding JAR packages and add them to the *classpath* environment variable of the project.

Step 2 Modify the code for submission of the topology. The following uses WordCount as an example:

1. Keep the structure of the Storm topology unchanged, including the Spout and Bolt developed using Storm API.

```
TopologyBuilder builder = new TopologyBuilder();
builder.setSpout("spout", new RandomSentenceSpout(), 5);
builder.setBolt("split", new SplitSentenceBolt(), 8).shuffleGrouping("spout");
builder.setBolt("count", new WordCountBolt(), 12).fieldsGrouping("split", new Fields("word"));
```

2. Modify the code for submission of the topology. An example is described as follows:

```
Config conf = new Config();
conf.setNumWorkers(3);
StormSubmitter.submitTopology("word-count", conf, builder.createTopology());
```

Perform the following operations:

```
Config conf = new Config();
 conf.setNumWorkers(3);
 //converts Storm Config to StormConfig of Flink.
 StormConfig stormConfig = new StormConfig(conf);
 //Construct FlinkTopology using TopologBuilder of Storm.
 FlinkTopology topology = FlinkTopology.createTopology(builder);
 //Obtain the Stream execution environment.
 StreamExecutionEnvironment env = topology.getExecutionEnvironment();
 //Set StormConfig to the environment variable of Job to construct Bolt and Spout.
 //If StormConfig is not required during the initialization of Bolt and Spout, you do not need to set this
parameter.
 env.getConfig().setGlobalJobParameters(stormConfig);
 //Submit the topology.
 topology.execute();
```

3. After the package is repacked, run the following command to submit the package:

flink run -class {MainClass} WordCount.jar

----End

28.9.3 Performing Embedded Service Migration

Scenarios

This section describes how to embed Storm code in DataStream of Flink in embedded migration mode. For example, the code of Spout or Bolt compiled using Storm API is embedded.

Procedure

Step 1 In Flink, perform embedded conversion to Spout and Bolt in the Storm topology to convert them to Flink operators. The following is an example of the code:

```
//set up the execution environment
final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
//get input data
final DataStream<String> text = getTextDataStream(env);
final DataStream<Tuple2<String, Integer>> counts = text
  //split up the lines in pairs (2-tuples) containing: (word,1)
  //this is done by a bolt that is wrapped accordingly
  .transform("CountBolt",
    TypeExtractor.getForObject(new Tuple2<String, Integer>("", 0)),
    new BoltWrapper<String, Tuple2<String, Integer>>(new CountBolt()))
  //group by the tuple field "0" and sum up tuple field "1"
  .keyBy(0).sum(1);
// execute program
env.execute("Streaming WordCount with bolt tokenizer");
```

Step 2 After the modification, run the following command to submit the modification:

flink run -class {MainClass} WordCount.jar

----End

28.9.4 Migrating Services of External Security Components Interconnected with Storm

Migrating Services for Interconnecting Storm with HDFS and HBase

If the Storm services use the **storm-hdfs** or **storm-hbase** plug-in package for interconnection, you need to specify the following security parameters when migrating Storm services as instructed in **Completely Migrating Storm Services**.

```
//Initialize Storm Config.
Config conf = new Config();

//Initialize the security plug-in list.
List<String> auto_tgts = new ArrayList<String>();
//Add the AutoTGT plug-in.
auto_tgts.add("org.apache.storm.security.auth.kerberos.AutoTGT");
//Add the AutoHDFS plug-in.
//If HBase is interconnected, use auto_tgts.add("org.apache.storm.hbase.security.AutoHBase") to replace the following:
auto_tgts.add("org.apache.storm.hdfs.common.security.AutoHDFS");

//Set security parameters.
conf.put(Config.TOPOLOGY_AUTO_CREDENTIALS, auto_tgts);
//Set the number of workers.
conf.setNumWorkers(3);

//Convert Storm Config to StormConfig of Flink.
StormConfig stormConfig = new StormConfig(conf);

//Construct FlinkTopology using TopologBuilder of Storm.
FlinkTopology topology = FlinkTopology.createTopology(builder);

//Obtain the StreamExecutionEnvironment.
StreamExecutionEnvironment env = topology.getExecutionEnvironment();

//Add StormConfig to the environment variable of Job to construct Bolt and Spout.
//If Config is not required during the initialization of Bolt and Spout, do not set this parameter.
env.getConfig().setGlobalJobParameters(stormConfig);

//Submit the topology.
topology.execute();
```

After the preceding security plug-in is configured, unnecessary logins during the initialization of HDFSBolt and HBaseBolt are avoided because the security context has been configured in Flink.

Migrating Services of Storm Interconnected with Other Security Components

If the plug-in packages, such as **storm-kakfa-client** are used for interconnection between Storm and other components for service migration, the security plug-ins configured previously need to be deleted.

```
List<String> auto_tgts = new ArrayList<String>();
//keytab mode
auto_tgts.add("org.apache.storm.security.auth.kerberos.AutoTGTFromKeytab");

//Write the plug-in list configured on the client to the specified config parameter.
```

```
//Mandatory in security mode
//This configuration is not required in common mode, and you can comment out the following line.
conf.put(Config.TOPOLOGY_AUTO_CREDENTIALS, auto_tgts);
```

The AutoTGTFromKeytab plug-in must be deleted during service migration. Otherwise, the login will fail when Bolt or Spout is initialized.

28.10 Storm Log Introduction

This section applies to MRS 3.*x* or later.

Log Description

Log paths: The default paths of Storm log files are **/var/log/Bigdata/storm/***Role name* (run logs) and **/var/log/Bigdata/audit/storm/***Role name* (audit logs).

- Nimbus: **/var/log/Bigdata/storm/nimbus** (run logs) and **/var/log/Bigdata/audit/storm/nimbus** (audit logs)
- Supervisor: **/var/log/Bigdata/storm/supervisor** (run logs) and **/var/log/Bigdata/audit/storm/supervisor** (audit logs)
- UI: **/var/log/Bigdata/storm/ui** (run logs) and **/var/log/Bigdata/audit/storm/ui** (audit logs)
- Logviewer: **/var/log/Bigdata/storm/logviewer** (run logs) and **/var/log/Bigdata/audit/storm/logviewer** (audit logs)

Log archive rule: The automatic Storm log compression function is enabled. By default, when the size of logs exceeds 10 MB, logs are automatically compressed into a log file named in the following format: *<Original log name>***.log.***[ID]***.gz**. A maximum of 20 latest compressed files are reserved by default. You can configure the number of compressed files and the compression threshold.

Names of compressed audit log files are in the format of **audit.log.***[yyyy-MM-dd]***.***[ID]***.zip**. These files permanently exist.

Table 28-2 Storm log list

Log Type	Log File Name	Description
Run log	nimbus/access.log	Nimbus user access log
	nimbus/nimbus-<PID>-gc.log	GC log of the Nimbus process
	nimbus/checkavailable.log	Nimbus availability check log
	nimbus/checkService.log	Nimbus serviceability check log
	nimbus/metrics.log	Nimbus monitoring statistics log
	nimbus/nimbus.log	Run log of the Nimbus process

Log Type	Log File Name	Description
	nimbus/postinstall.log	Work log after Nimbus installation
	nimbus/prestart.log	Work log before Nimbus startup
	nimbus/start.log	Work log of Nimbus startup
	nimbus/stop.log	Work log of Nimbus shutdown
	supervisor/access.log	Supervisor access log
	supervisor/metrics.log	Supervisor monitoring statistics log
	supervisor/postinstall.log	Work log after supervisor installation
	supervisor/prestart.log	Work log before supervisor startup
	supervisor/start.log	Work log of supervisor startup
	supervisor/stop.log	Work log of supervisor shutdown
	supervisor/supervisor.log	Run log of the supervisor process
	supervisor/supervisor-<PID>-gc.log	GC log of the supervisor process
	ui/access.log	UI access log
	ui/metric.log	UI monitoring statistics log
	ui/ui-<PID>-gc.log	GC log of the UI process
	ui/postinstall.log	Work log after UI installation
	ui/prestart.log	Work log before UI startup
	ui/start.log	Work log of UI startup
	ui/stop.log	Work log of UI shutdown
	ui/ui.log	Run log of the UI process
	logviewer/access.log	Logviewer access log

Log Type	Log File Name	Description
	logviewer/metric.log	Logviewer monitoring statistics log
	logviewer/logviewer-<PID>-gc.log	GC log file of the logviewer process
	logviewer/logviewer.log	Run log of the logviewer process
	logviewer/postinstall.log	Work log after logviewer installation
	logviewer/prestart.log	Work log before logviewer startup
	logviewer/start.log	Work log of logviewer startup
	logviewer/stop.log	Work log of logviewer shutdown
	supervisor/[topologyId]-worker-[*Port number*].log	Run log of the Worker process. One port occupies one log file. By default, the system contains five ports: 29100, 29101, 29102, 29103 and 29304.
	supervisor/metadata/[topologyid]-worker-[*Port number*].yaml	Worker log metadata file, which is used by logviewer to delete logs. This file is automatically deleted by the logviewer log deletion thread based on certain conditions.
	nimbus/cleanup.log	Cleanup log of Nimbus uninstallation
	logviewer/cleanup.log	Cleanup log of logviewer uninstallation
	ui/cleanup.log	Cleanup log of UI uninstallation
	supervisor/cleanup.log	Cleanup log of supervisor uninstallation
	leader_switch.log	Run log file that records the Storm active/standby switchover
Audit log	nimbus/audit.log	Nimbus audit log

Log Type	Log File Name	Description
	ui/audit.log	UI audit log
	supervisor/audit.log	Supervisor audit log
	logviewer/audit	Logviewer audit log

Log Levels

Table 28-3 describes the log levels supported by Storm.

Levels of run logs and audit 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 28-3 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:

Step 1 Go to the **All Configurations** page of Storm 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

Log Format

The following table lists the Storm log formats:

Table 28-4 Log Formats

Log Type	Format	Example
Run log	%d{yyyy-MM-dd HH:mm:ss,SSS} | %-5p | [%t] | %m | %logger (%F:%L) %n	2015-03-11 23:04:00,241 | INFO | [RMI TCP Connection(2646)-10.0.0.2] | The baseSleepTimeMs [1000] the maxSleepTimeMs [1000] the maxRetries [1] | backtype.storm.utils.StormBoundedExponential-BackoffRetry (StormBoundedExponentialBackoffRetry.java:46)
	<yyyy-MM-dd HH:mm:ss,SSS><HostName><RoleName><logLevel><Message>	2017-03-28 02:57:52 493 10-5-146-1 storm- INFO Nimbus start normally
Audit log	*<Username><User IP address><Time><Operation><Operation object><Operation result>*	UserName=storm/hadoop, UserIP=10.10.0.2, Time=Tue Mar 10 01:15:35 CST 2015, Operation=Kill, Resource=test, Result=Success

28.11 Performance Tuning

28.11.1 Storm Performance Tuning

Scenario

You can modify Storm parameters to improve Storm performance in specific service scenarios.

This section applies to MRS 3.x or later.

Modify the service configuration parameters. For details, see **Modifying Cluster Service Configuration Parameters**.

Topology Tuning

This task enables you to optimize topologies to improve efficiency for Storm to process data. It is recommended that topologies be optimized in scenarios with lower reliability requirements.

Table 28-5 Tuning parameters

Parameter	Default Value	Scenario
topology.acker.executors	null	Specifies the number of acker executors. If a service application has lower reliability requirements and certain data does not need to be processed, this parameter can be set to **null** or **0** so that you can set acker off, flow control is weakened, and message delay is not calculated. This improves performance.
topology.max.spout.pending	null	Specifies the number of messages cached by spout. The parameter value takes effect only when acker is not **0** or **null**. Spout adds each message sent to downstream bolt into the pending queue. The message is removed from the queue after downstream bolt processes the message and the processing is confirmed. When the pending queue is full, spout stops sending messages. Increasing the pending value improves the message throughput of spout per second but prolongs the delay.
topology.transfer.buffer.size	32	Specifies the size of the Distuptor message queue for each worker process. It is recommended that the size be between 4 to 32. Increasing the queue size improves the throughput but may prolong the delay.
RES_CPUSET_PERCENTAGE	80	Specifies the percentage of physical CPU resources used by the supervisor role instance (including startup and management worker processes) on each node. Adjust the parameter value based on service volume requirements of the node on which the supervisor exists, to optimize CPU usage.

JVM Tuning

If an application must occupy more memory resources to process a large volume of data and the size of worker memory is greater than 2 GB, the G1 garbage collection algorithm is recommended.

Table 28-6 Tuning parameters

Parameter	Default Value	Scenario
WORKER_GC_OPTS	-Xms1G -Xmx1G -XX:+UseG1GC -XX:+PrintGCDetails -Xloggc:artifacts/gc.log -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=artifacts/heapdump	If an application must occupy more memory resources to process a large volume of data and the size of worker memory is greater than 2 GB, the G1 garbage collection algorithm is recommended. In this case, change the parameter value to **-Xms2G -Xmx5G -XX:+UseG1GC**.

29 Using Tez

29.1 Accessing the Tez Web UI to View the Task Execution Result

Tez displays the Tez task execution process on a GUI. You can view the task execution details on the GUI.

This section applies to MRS 3.x and later versions.

Prerequisite

The TimelineServer instance of the Yarn service has been installed.

How to Use

Log in to Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. 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.

29.2 Common Tez Parameters

Navigation path for setting parameters:

On Manager, choose **Cluster** > **Service** > **Tez** > **Configuration** > **All Configurations**. Enter a parameter name in the search box.

This section applies to MRS 3.x and later versions.

Parameter description

Table 29-1 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

29.3 Log Overview

This section applies to MRS 3.x and later versions.

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 29-2 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

Log Type	Name	Description
	startDetail.log	Startup log of the TezUI process
	stopDetail.log	Stop log of the TezUI process
Audit log	tezui-audit.log	TezUI audit log

Log Level

Table 29-3 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 29-3 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 29-4 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)

29.4 Common Issues

29.4.1 TezUI Cannot Display Tez Task Execution Details

Question

After a user logs in to Manager and switches to the Tez web UI, the submitted Tez tasks are not displayed.

Answer

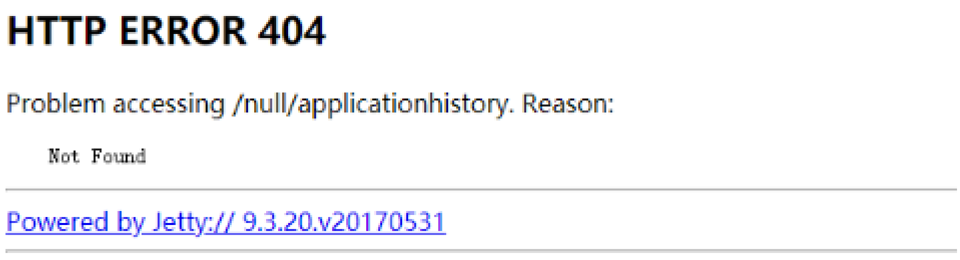
The Tez task data displayed on the Tez WebUI requires the support of TimelineServer of Yarn. Ensure that TimelineServer has been enabled and is running properly before the task is submitted.

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**.

29.4.2 Error Occurs When a User Switches to 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.



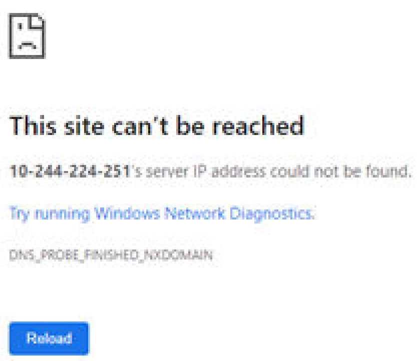
Answer

The Tez web UI depends on the TimelineServer instance of Yarn. Therefore, TimelineServer must be installed in advance and in the **Good** state.

29.4.3 Yarn Logs Cannot Be Viewed on the TezUI Page

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. Perform the following steps:

Modify the **C:\Windows\System32\drivers\etc\hosts** file on the Windows host and add a line indicating the mapping between the host name and IP address, for example, **10.244.224.45 10-044-224-45**. Save the modification and access the host again.

29.4.4 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:

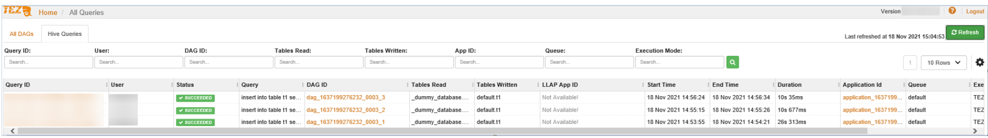
On FusionInsight Manager, choose **Cluster** > **Service** > **Hive** and click the **Configurations** tab and then **All Configurations**. In the navigation pane on the left, choose **HiveServer** > **Customization**. Add the following configuration to **hive-site.xml**:

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.



30 Using YARN

30.1 YARN User Permission Management

30.1.1 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** for clusters of MRS 3.*x* or later.

Procedure

For versions earlier than MRS 3.x, perform the following operations:

Step 1 Log in to Manager and choose **System** > **Manage Role** > **Create Role**.

Step 2 Click **Create Role** and fill in **Role Name** and **Description**.

Step 3 Set permissions. For details, see **Table 30-1**.

Yarn permissions:

- **Cluster Admin Operations**: Yarn administrator permissions.
- **Scheduler Queue**: queue resources management .

Table 30-1 Setting a role

Task	Operation
Setting the Yarn administrator permission	In the **Permission** table, click **Yarn** and select **Cluster Admin Operations**. **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 **Permission** table, choose **Yarn** > **Scheduler Queue**. 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 **Permission** table, choose **Yarn** > **Scheduler Queue**. 2. In the **Permission** column of the specified queue, select **Admin**.

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

For MRS 3.*x* or later, perform the following operations:

Step 1 Log in to Manager and choose **System** > **Permission** > **Role**.

Step 2 Click **Create Role** and set a role name and enter description.

Step 3 Refer **Table 30-2** to configure resource permissions for roles.

Yarn permissions:

- Cluster management: Yarn administrator permissions.
- Queue scheduling: queue resource management.

Table 30-2 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

30.2 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

- You have installed the client.

  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 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. In common mode, you do not need to download the keytab file or change the password.

Using the Yarn 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 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 5 Run the Yarn command. The following provides an example:

yarn application -list

----End

Client-related FAQs

- Question 1: What do I do when the HDFS client exits abnormally and error message "java.lang.OutOfMemoryError" is displayed during client command execution?

  Memory required for running the Yarn client exceeds the upper limit (128 MB by default) set on the Yarn client.

  a. You can change the maximum memory of the Yarn client by modifying parameters in *<Client installation path>***/HDFS/component_env**.

     - For MRS 3.x and later versions, modify the **CLIENT_GC_OPTS** parameter. For example, to set the maximum memory to 1 GB, run the following command:
       ```
       export CLIENT_GC_OPTS="-Xmx1G"
       ```
     - For MRS 3.x and earlier versions, modify the **GC_OPTS_YARN** parameter. For example, to set the maximum memory to 1 GB, run the following command:
       ```
       export GC_OPTS_YARN="-Xmx1G"
       ```

  b. After the modification, run the following command to make the modification take effect:

     source *<Client installation path>*/**/bigdata_env**

- Question 2: How do I set the run log level of the Yarn client?

  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

30.3 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 30-3** 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 30-5** describes the configurations in the *Client installation path***/Yarn/config/mapred-site.xml** configuration file on the MapReduce client node.

Table 30-3 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. ● For clusters of versions earlier than MRS 3.x: The valid value range of the maximum disk quota percentage is 0 to 100. If the value is less than or equal to **0**, it is forcibly reset to **25**. If the value is greater than 100, the value is forcibly reset to **25**. ● For clusters of MRS 3.*x* or later: 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

Parameter	Description	Default Value
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
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 30-4** in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 30-4 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 30-5** in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 30-5 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

30.4 Enabling Yarn CGroups to Limit the Container CPU Usage

Scenario

CGroups is a Linux kernel feature. It groups task sets and their subsets into hierarchical structures with customizable behavior. In Yarn, this feature allows you to easily limit the resources (such as CPU) used by containers.

NOTE

Currently, CGroups is only used for limiting the CPU usage.

This section is available for clusters of MRS 3.*x* and later.

Configuration Description

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:

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 CPUset mode can be used for Linux versions, for example, Cent OS, that do not support the Quota mode.

The table below describes the parameter for configuring cpuset mode, that is, only configured CPUs can be used by Yarn. Add the following parameters on Manager.

Table 30-6 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

The table below describes the parameter for configuring strictcpuset mode, that is, only configured CPUs can be used by container. Add the following parameters on Manager.

Table 30-7 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 only. Value **true** indicates that the container can only use the allocated CPUs.	false

To switch from cpuset mode to quota mode, the following conditions must be met:

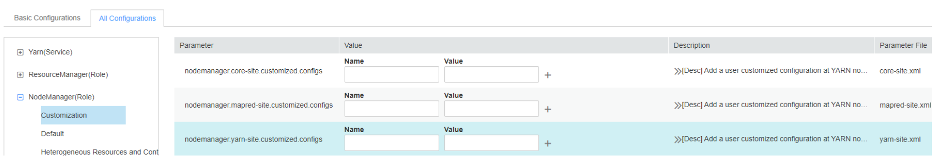
- Set the **yarn.nodemanager.linux-container-executor.cgroups.cpu-set-usage** parameter to **false**.
- Delete the **container** folder (if any) from the **/sys/fs/cgroup/cpuset/hadoop-yarn/** directory.
- Delete all CPUs configured in the **cpuset.cpus** file in **/sys/fs/cgroup/cpuset/hadoop-yarn/**.

Procedure

Step 1 Log in to Manager. Choose **Cluster** > **Services** > **Yarn**. Click **Configurations** and click **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 30-6** and **Table 30-7** 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

30.5 Enterprise-Class Enhancement of YARN

30.5.1 Configuring the Yarn Permission Control

Scenario

In a multi-tenant security mode, multiple users can share a cluster, submit, and execute their tasks without seeing each other's information. To prevent users from accessing each other's task details, a permission control system is necessary.

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 30-8** in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 30-8 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**. **NOTE** This parameter applies to clusters of MRS 3.*x* or later.	true

- Viewing MapReduce configuration parameters

  Go to the **All Configurations** page of MapReduce and search for a parameter name in **Table 30-9**. For details, see **Modifying Cluster Service Configuration Parameters**.

Table 30-9 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**. **NOTE** This parameter applies to clusters of MRS 3.*x* or later.	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 **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.

30.5.2 Specifying the User Who Runs Yarn Tasks

This section applies to clusters of MRS 3.*x* or later.

Scenario

Yarn allows the user who starts NodeManager to run tasks submitted by all users, and also allows the user who submits a task to run it.

Configuration Description

Log in to Manager, choose **Cluster** > **Services** > **Yarn** and click **Configurations** then **All Configurations**. Enter a parameter name in the search box.

Table 30-10 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. An empty value (default value) indicates that the user running the container is also the one submitting the task. 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**.

30.5.3 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 30-11** in the search box by referring to **Modifying Cluster Service Configuration Parameters**.

Table 30-11 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

30.5.4 Configure the ApplicationMaster to Automatically Adjust the Allocated Memory

This section applies to clusters of MRS 3.*x* or later.

Scenario

When the ApplicationMaster creates a container, it automatically adjusts the allocated memory based on the total number of tasks, making resource utilization more flexible and improving client application fault tolerance.

Configuration

Navigation path for setting parameters:

Go to the **All Configurations** page of Yarn and enter **mapreduce.job.am.memory.policy** in the search box. For details, see **Modifying Cluster Service Configuration Parameters**.

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 30-12 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 (the sum of the Map and Reduce tasks) 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.
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

Parameter	Description	Setting Requirement
maxMemory	Indicates the upper limit of memory that can be automatically adjusted. If the memory exceeds the upper limit, this value will be used.	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**.

30.5.5 Configuring ApplicationMaster Work Preserving

This section applies to clusters of MRS 3.*x* or later.

Scenario

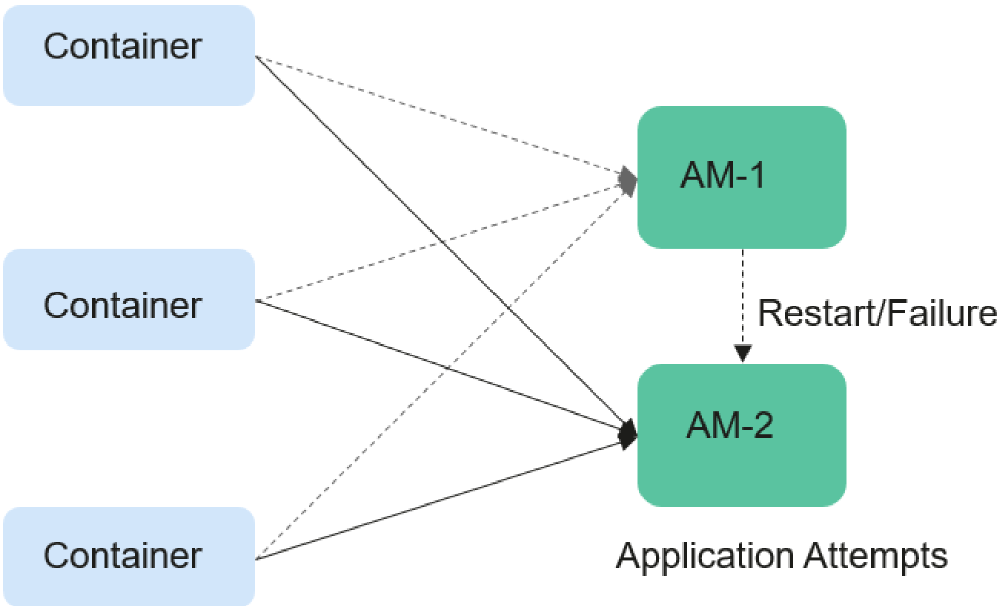
In YARN, ApplicationMasters run on NodeManagers just like every other container (ignoring unmanaged ApplicationMasters in this context). ApplicationMasters may break down, exit, or shut down. If an ApplicationMaster node goes down, ResourceManager kills all the containers of ApplicationAttempt, including 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 30-1 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 30-13**.

Table 30-13 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
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

Parameter	Description	Default Value
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

30.5.6 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

30.5.7 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 30-14 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.	-

30.5.8 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.

Procedure

For versions earlier than MRS 3.*x*, perform the following operations:

Step 1 Choose **Clusters** > **Active Clusters**, and click a cluster name. Choose **Components** > **Yarn** > **Instances**.

Step 2 Click **NodeManager** in the **Role** column and go to the **Instance Configuration** tab page. Select **All** from the **Basic** drop-down list, and search for the required parameters.

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% to 90% 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 Configuration**, select **Restart the affected services or instances**, and click **OK** to restart the NodeManager role instance.

Operation succeeded is displayed. Click **Finish**. The NodeManager role instance is started successfully.

----End

For MRS 3.*x* or later, perform the following operations:

Step 1 Log in to FusionInsight Manager and choose **Cluster**, click the name of the desired cluster, and choose **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

30.5.9 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 30-15 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

Parameter	Description	Default Value
yarn.resourcemanager.store.class	State-store class used to store the application and task statuses and certificate content.	For clusters of versions earlier than MRS 3.x: **org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore** For clusters of MRS 3.*x* or later: 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.	For clusters of versions earlier than MRS 3.x: **false** For MRS 3.x or later: **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.	For clusters of versions earlier than MRS 3.x: **1** For MRS 3.x or later: **20**

Configure NodeManager Restart as follows:

Table 30-16 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 It applies to clusters of MRS 3.*x* or later.	${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

30.6 Yarn Performance Tuning

30.6.1 Preempting a Task

Scenario

The capacity scheduler of ResourceManager implements job preemption to simplify job running in queues and improve resource utilization. The process is as follows:

1. 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%.
2. 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.
3. Task 1 uses 25% cluster resources provided by Queue A and 50% resources from Queue B. Queue B reserves 25% cluster resources.
4. 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.
5. Task 1 will be executed when Task 2 is complete and the cluster has sufficient resources.

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 30-17 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

Parameter	Description	Default Value
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
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

Parameter	Description	Default Value
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

30.6.2 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)**.

30.6.3 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. If the total physical memory space of a node is large and there is no resident process of other services, set this parameter to the total physical memory minus the memory occupied by the resident processes of NodeManager.	MRS 3.*x* or later: **16384** Versions earlier than MRS 3.x: **8192**

- **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. ● If the task is computing-intensive, set this parameter to the number of logical CPU cores. ● If the task is not computing-intensive, set this parameter to 1.5 to 2 times the number of logical CPU cores. ● If the number of CPU cores used by a task differs greatly from the memory resources, configure the CPU resources based on the memory resources. For example, most tasks use one core and 3 GB memory. If **yarn.nodemanager.resource.memory-mb** is 380 GB, set this parameter to **128**.	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 need 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 need 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.	

30.7 YARN O&M Management

30.7.1 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:

For versions earlier than MRS 3.x, perform the following operations:

On the MRS console, choose **Tenants** > **Resource Distribution Policies**.

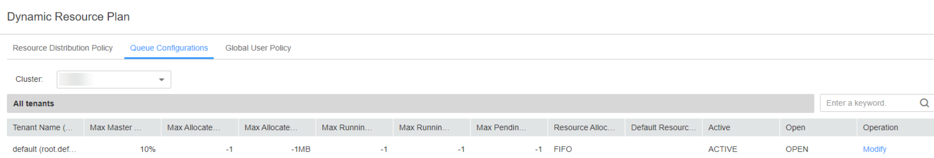
The following uses the **default** queue as an example. The configurations of other queues are similar. Click **Modify** to edit the parameters.

Table 30-18 Parameter description

Parameter	Description	Default Value
Capacity	Queue resource capacity (percentage). Ensure that the capacity requirement of each queue is satisfied when the system is busy. If only a few application programs are running in a queue, the remaining resource of the queue can be shared with other queues. Note that the total capacity of all queues must be smaller than 100.	20
Maximum Capacity	Maximum queue resource usage (percentage). Due to resource sharing, the resources used by a queue may exceed its capacity. The maximum resource usage can be limited using this parameter.	100

For MRS 3.*x* or later, perform the following operations:

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 30-19 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.

Parameter	Description
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.
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 30-20**. For details, see **Modifying Cluster Service Configuration Parameters**.

Table 30-20 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 30-21**. For details, see **Modifying Cluster Service Configuration Parameters**.

Table 30-21 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

Parameter	Description	Default Value
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.

30.7.2 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 30-22 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

Log Type	Log File Name	Description
	<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
	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

Log Type	Log File Name	Description
	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 30-23 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 30-23 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.
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 30-24 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*>	2014-09-26 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*>	2014-09-26 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)

30.7.3 Configuring the Localized Log Levels

This section applies to clusters of MRS 3.*x* or later.

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 30-25 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.

30.7.4 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 30-26 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.	For versions earlier than MRS 3.x: false For MRS 3.x or later: 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

30.7.5 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

For versions earlier than MRS 3.*x*, perform the following operations:

Step 1 Check the environment.

1. Log in to the MRS console. In the left navigation pane, choose **Clusters** > **Active Clusters**, and click a cluster name. Choose **Components** and check whether health status of Yarn is **Good**.
   – If yes, go to **Step 1.3**.
   – If no, the Yarn status is unhealthy. Go to **Step 1.2**.
2. Rectify the Yarn fault. 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 **Step 2**.
   – To change the storage directory of a single NodeManager instance, go to **Step 3**.

Step 2 Change the storage directory of the NodeManager role.

1. Choose **Clusters** > **Active Clusters**, and click a cluster name. Choose **Components** > **Yarn** > **Stop** to stop the Yarn service.
2. Log in to the ECS server and go to each node where Yarn is installed as user **root**. 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 MRS console, choose **Clusters** > **Active Clusters** and click a cluster name. Choose **Components** > **Yarn** > **Instances**. Select the NodeManager instance of the corresponding host. Choose **Instance Configuration** > **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 Configuration**, select **Restart the affected services or instances**, and 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 **Clusters** > **Active Clusters**, and click a cluster name. Choose **Components** > **Yarn** > **Instances**. Select the NodeManager instance whose storage directory needs to be modified, and choose **More** > **Stop Instance**.
2. Log in to the ECS and go to the NodeManager node as user **root**. 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 MRS console, click the specified NodeManager instance and switch to the **Instance Configuration** tab 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 Configuration** and select **Restart the affected services or instances**. Click **OK** to restart the NodeManager instance.

   Click **Finish** when the system displays "Operation successful". The NodeManager instance is successfully started.

----End

For MRS 3.*x* or later, perform the following operations:

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

30.8 Common Issues About Yarn

30.8.1 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.

30.8.2 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.

30.8.3 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.

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.

30.8.4 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.

30.8.5 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.

30.8.6 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 resource pool scenarios. You can change the value of **yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold** to **(nodes number of the pool / total nodes) x 33%** to solve this problem. The path is *Client installation path*/**Yarn/config/yarn-site.xml**.

30.8.7 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.

30.8.8 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**.

30.8.9 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.

30.8.10 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:

In the *Client installation path***/Yarn/config/yarn-site.xml** file, modify the **yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold** parameter to configure the threshold (in percentage) for ignoring blacklisted nodes. 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.

30.8.11 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.

30.8.12 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 30-27 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

30.8.13 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 30-28 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

Parameter	Description	Default Value	Default value and default permissions on the parent directory
mapreduce.jobhistory.done-dir	Directory of historical files managed by the MR JobHistory Server.	${yarn.app.mapreduce.am.staging-dir}/history/done	777

30.8.14 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**.

31 Using ZooKeeper

31.1 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

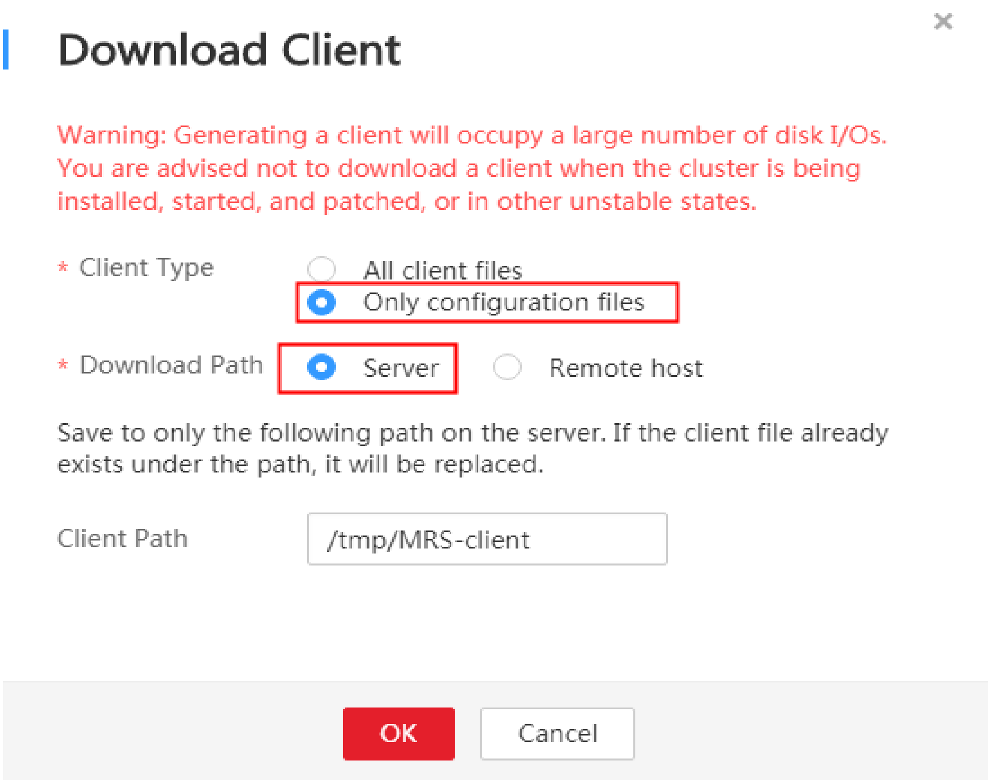
For MRS 2.*x* or earlier, perform the following operations:

Step 1 Download the client configuration file.

1. Log in to the MRS console. In the left navigation pane, choose **Clusters** > **Active Clusters**, and click the cluster to be operated.
2. Click the **Components** tab.
3. Click **Download Client**.

   Set **Client Type** to **Only configuration files**, **Download Path** to **Server**, and click **OK** to generate the client configuration file. The generated file is saved in the **/tmp/MRS-client** directory on the active management node by default.

Figure 31-1 Downloading only the client configuration files



Step 2 Log in to the active management node of MRS Manager.

1. On the MRS console, choose **Clusters > Active Clusters** and click a cluster name. On the **Nodes** tab, view the node names. The node whose name contains **master1** is the Master1 node, and the node whose name contains **master2** is the Master2 node.

   The active and standby management nodes of MRS Manager are installed on Master nodes by default. Because Master1 and Master2 are switched over in active and standby mode, Master1 is not always the active management node of MRS Manager. Run a command in Master1 to check whether Master1 is active management node of MRS Manager. For details about the command, see **Step 2.4**.

2. Log in to the Master1 node using the password as user **root**.
3. Run the following commands to switch to user **omm**:

   sudo su - root

   su - omm

4. Run the following command to check the active management node of MRS Manager:

   sh ${BIGDATA_HOME}/om-0.0.1/sbin/status-oms.sh

   In the command output, the node whose **HAActive** is **active** is the active management node, and the node whose **HAActive** is **standby** is the standby management node. In the following example, **mgtomsdat-sh-3-01-1** is the active management node, and **mgtomsdat-sh-3-01-2** is the standby management node.

```
Ha mode
double
```

```
NodeName            HostName                  HAVersion         StartTime               HAActive
HAAllResOK          HARunPhase
192-168-0-30        mgtomsdat-sh-3-01-1       V100R001C01       2014-11-18 23:43:02
active              normal                    Actived
192-168-0-24        mgtomsdat-sh-3-01-2       V100R001C01       2014-11-21 07:14:02
standby             normal                    Deactived
```

5. Log in to the active management node, for example, **192-168-0-30** of MRS Manager 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/MRS-client/MRS_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.
```

NOTE

You can also refer to method 2 in **Updating a Client (Versions Earlier Than 3.x)** to perform operations in **Step 1** to **Step 4**.

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 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 *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***

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

For MRS 3.x or later, perform the following operations:

Step 1 Download the client configuration file.

1. Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**.
2. Choose **Cluster** > *Name of the desired cluster* > **Dashboard** > **More** > **Download Client**.
3. 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.. 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**

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

31.2 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 31-1 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:

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 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

31.3 Common ZooKeeper 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 31-2 Parameters

Parameter	Description	Default Value
skipACL	Specifies whether to skip the permission check of the ZooKeeper node.	no
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.

31.4 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 31-3 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

Log Type	Log File Name	Description
	zookeeper-period-check-java.log	ZooKeeper quota monitoring period check log
Audit Log	zk-audit-quorumpeer.log	ZooKeeper operation audit log

Log levels

Table 31-4 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 31-4 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 31-5 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)

31.5 Common Issues About ZooKeeper

31.5.1 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 31-6 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.

31.5.2 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 31-7 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

31.5.3 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.

31.5.4 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.

31.5.5 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";
};
```

31.5.6 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** file, set **useTicketCache** to **false**, set **useKeyTab** to **true**, and specify the keytab path.

31.5.7 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.

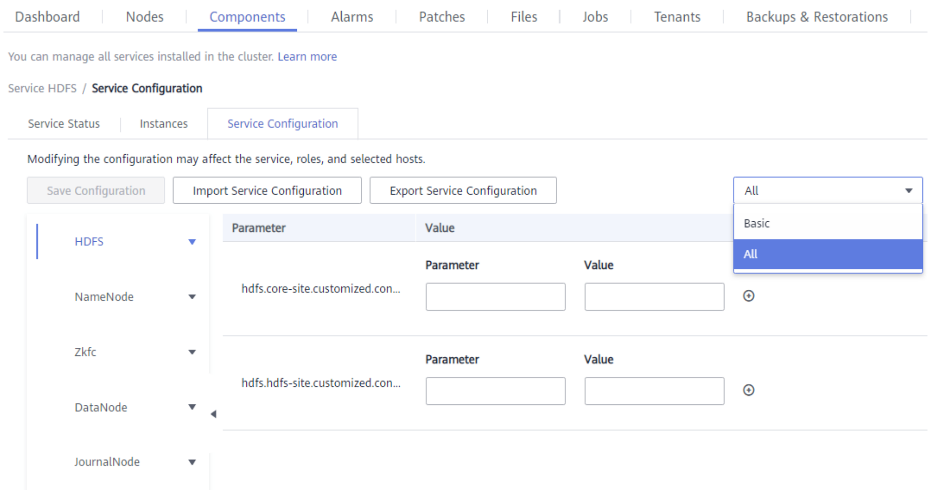
To delete the leader node, view the IP address of the node where the leader resides by referring to **How Do I Check Which ZooKeeper Instance Is a Leader?**, run the **zkCli.sh -server leader** *node IP address* **2181** command to connect to the client, and then run the **deleteall** command to delete the leader node. For details, see **Using ZooKeeper from Scratch**.

32 Appendix

32.1 Modifying Cluster Service Configuration Parameters

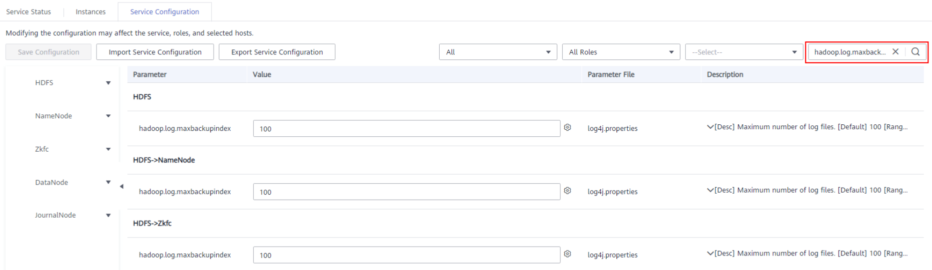
- You can modify service configuration parameters on the cluster management page of the MRS management console.
  a. Log in to the MRS console. In the left navigation pane, choose **Clusters** > **Active Clusters**, and click a cluster name.
  b. Choose **Components** > *Name of the desired service* > **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. .

- For MRS 3.*x* or earlier: You can log in to MRS Manager to modify service configuration parameters.

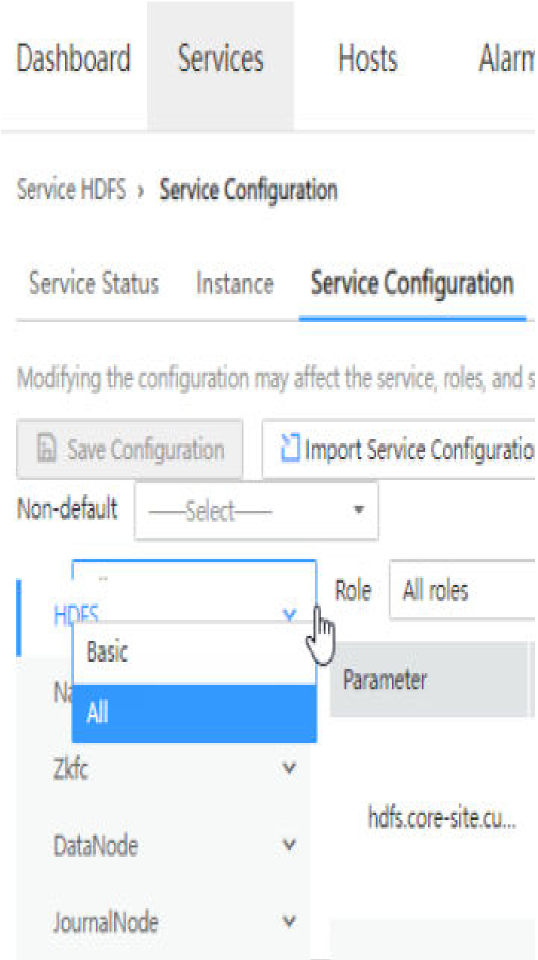
a. Log in to MRS Manager.

b. Click **Services**.

c. Click the specified service name on the service management page.

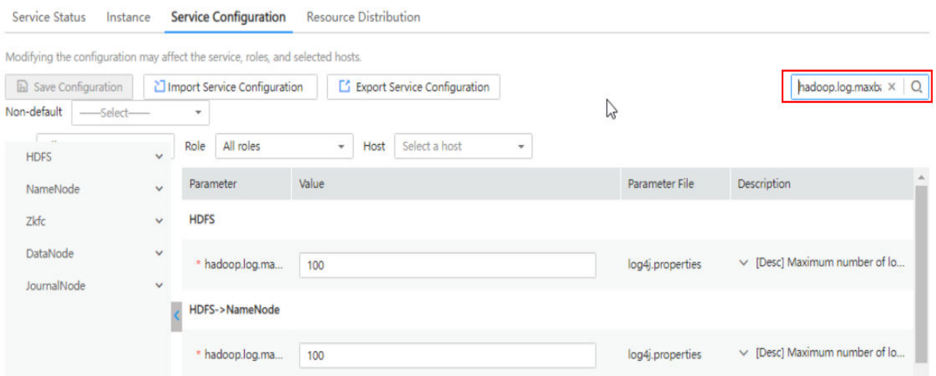
d. Click **Service Configuration**.

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.)



e. 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.)



f. Click **Save**. In the confirmation dialog box, click **OK**.

g. 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.

- For MRS 3.*x* or later: You can log in to FusionInsight Manager to modify service configuration parameters.

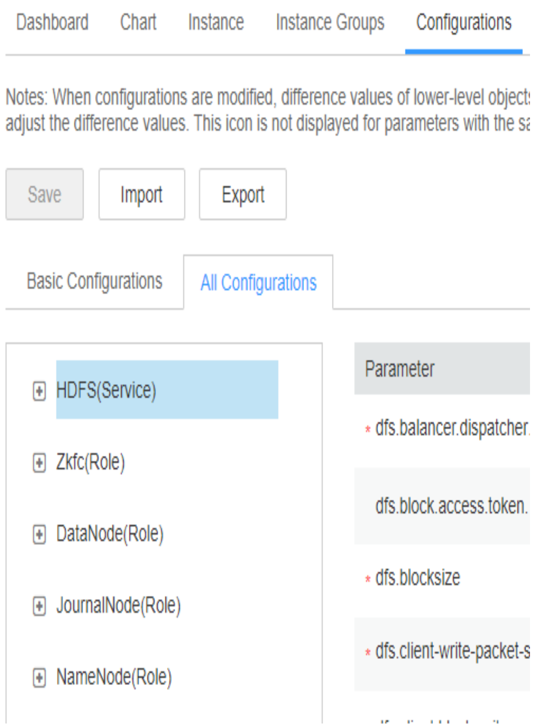
  a. You have logged in to FusionInsight Manager.

  b. Choose **Cluster** > **Service**.

  c. Click the specified service name on the service management page.

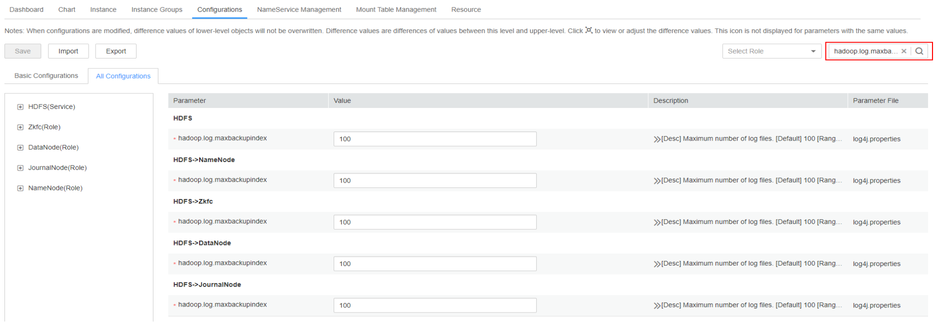
  d. Click **Configuration**.

     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.)



  e. 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.)



f. Click **Save**. In the confirmation dialog box, click **OK**.

g. 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.

32.2 Accessing Manager

32.2.1 Accessing MRS Manager (Versions Earlier Than MRS 3.x)

Scenario

Clusters of versions earlier than MRS 3.x use MRS Manager to monitor, configure, and manage clusters. You can open the MRS Manager page on the MRS console.

Accessing MRS manager

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 **Access Manager**. The **Access MRS Manager** page is displayed.

- If you have bound an EIP when creating a cluster, the page shown in **Figure 32-1** is displayed.

  a. Select the security group to which the security group rule to be added belongs. The security group is configured when the cluster is created.

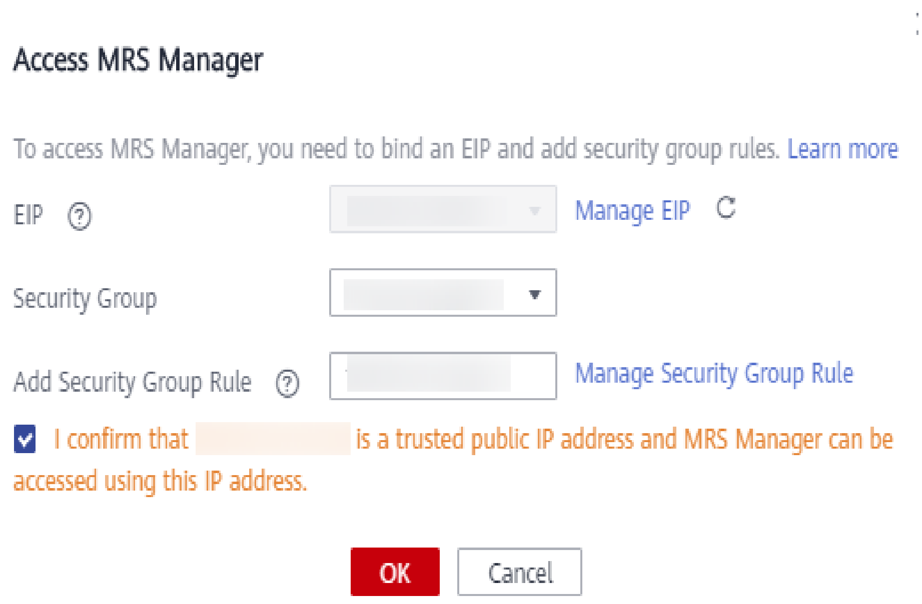
  b. Add a security group rule. By default, your public IP address used for accessing port 9022 is filled in the rule. To enable multiple IP address segments to access MRS Manager, see **Step 6** to **Step 9**. If you want to view, modify, or delete a security group rule, click **Manage Security Group Rule**.

NOTE

- It is normal that the automatically generated public IP address is different from the local IP address and no action is required.
- If port 9022 is a Knox port, you need to enable the permission of port 9022 to access Knox for accessing MRS Manager.

c. Select the checkbox stating that **I confirm that xx.xx.xx.xx is a trusted public IP address and MRS Manager can be accessed using this IP address**.

Figure 32-1 Adding a security group rule for accessing MRS Manager



- If you have not bound an EIP when creating a cluster, the page shown in **Figure 32-2** is displayed.

a. Select an available EIP from the drop-down list or click **Manage EIP** to buy one.

b. Select the security group to which the security group rule to be added belongs. The security group is configured when the cluster is created.

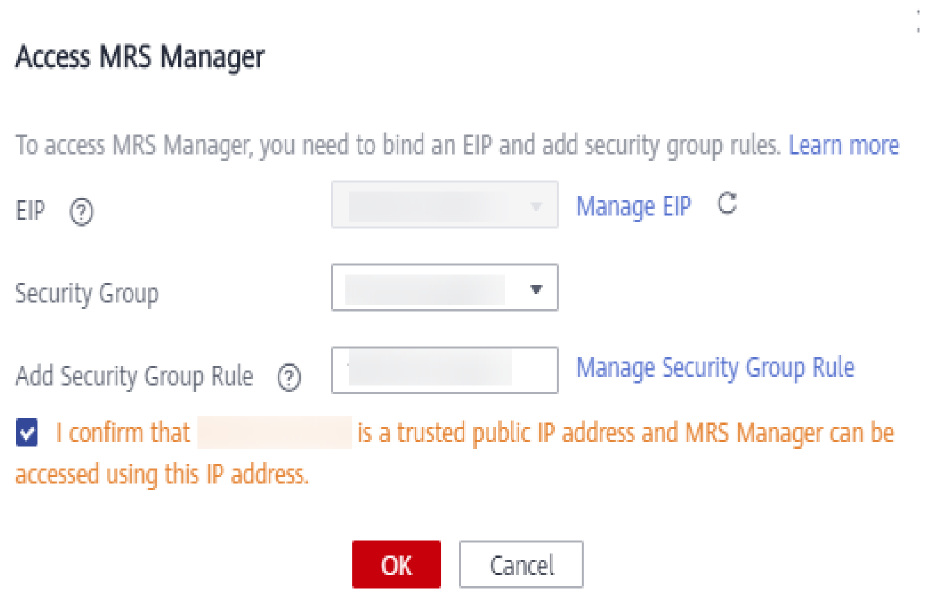
c. Add a security group rule. By default, your public IP address used for accessing port 9022 is filled in the rule. To enable multiple IP address segments to access MRS Manager, see **Step 6** to **Step 9**. If you want to view, modify, or delete a security group rule, click **Manage Security Group Rule**.

NOTE

- It is normal that the automatically generated public IP address is different from the local IP address and no action is required.
- If port 9022 is a Knox port, you need to enable the permission of port 9022 to access Knox for accessing MRS Manager.

d. Select the checkbox stating that **I confirm that xx.xx.xx.xx is a trusted public IP address and MRS Manager can be accessed using this IP address**.

Figure 32-2 Binding an EIP



Step 4 Click **OK**. The MRS Manager login page is displayed.

Step 5 Enter the default username **admin** and the password set during cluster creation, and click **Log In**. The MRS Manager page is displayed.

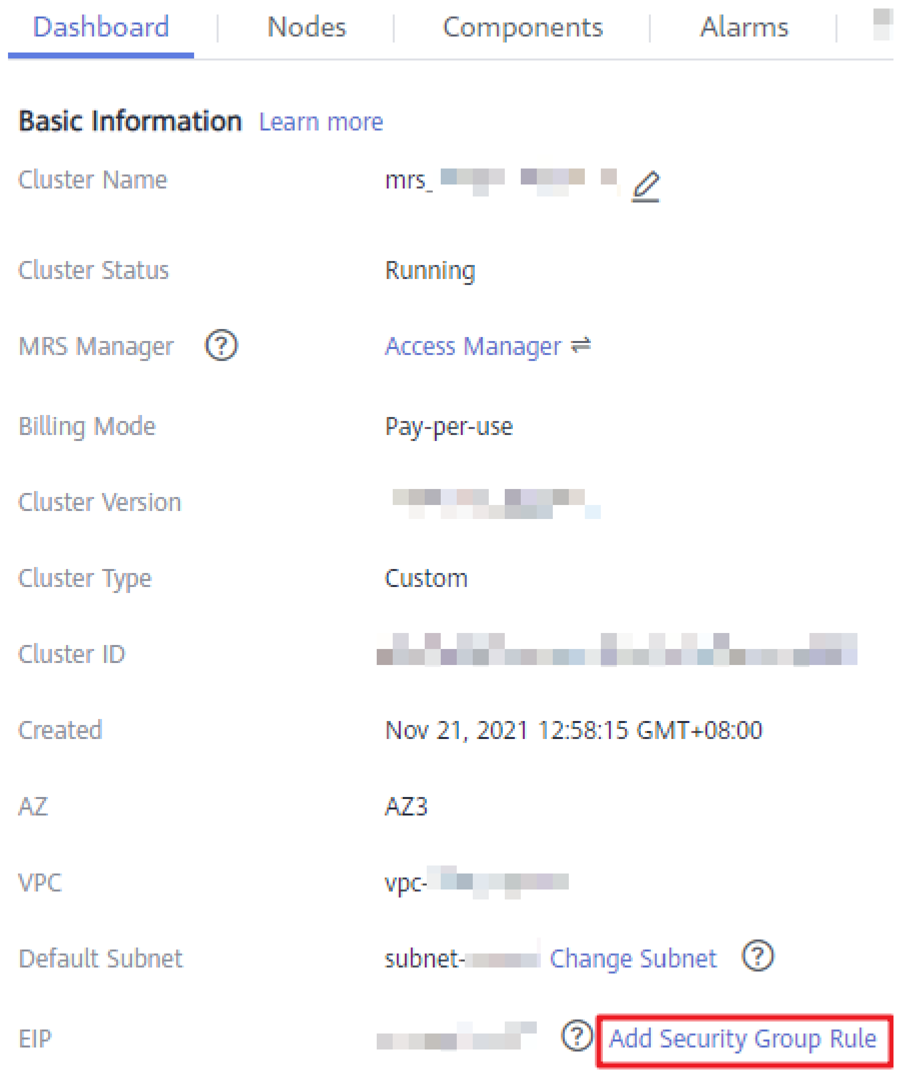
Step 6 On the MRS console, click **Clusters** and choose **Active Clusters**. Click the target cluster name to access the cluster details page.

NOTE

To assign MRS Manager access permissions to other users, follow instructions from **Step 6** to **Step 9** to add the users' public IP addresses to the trusted range.

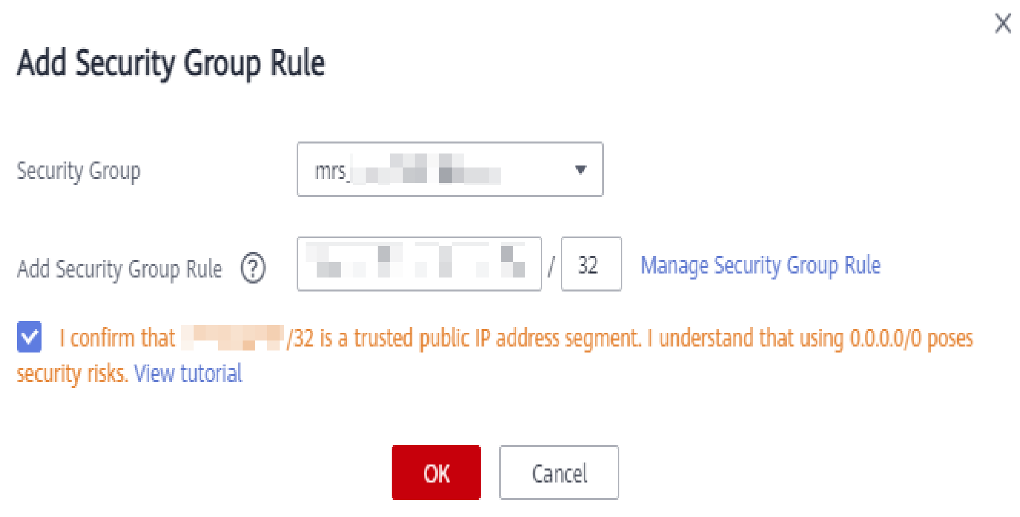
Step 7 Click **Add Security Group Rule** on the right of **EIP**. See **Figure 32-3**.

Figure 32-3 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 the authorized object is a trusted public IP address range. Do not use 0.0.0.0/0. Otherwise, security risks may arise**. See **Figure 32-4**.

Figure 32-4 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**.

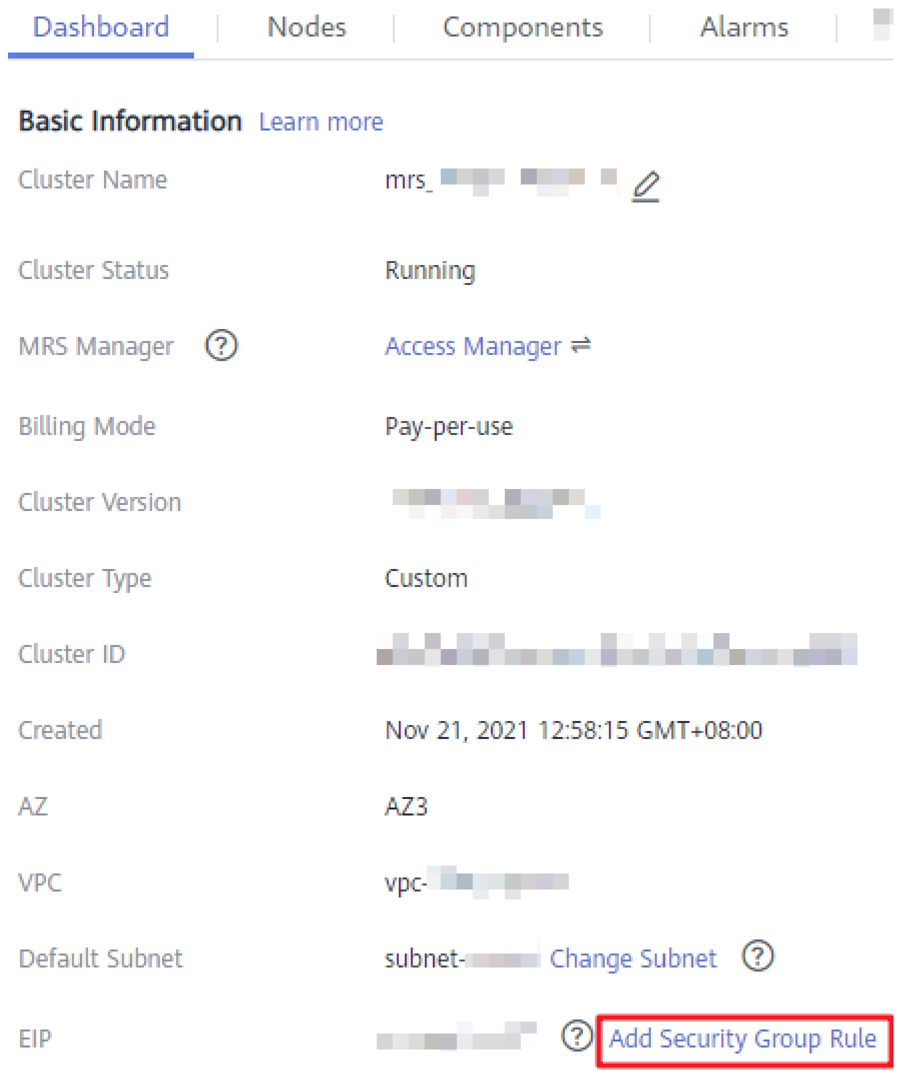
----End

Granting the Permission to Access MRS Manager to Other Users

Step 1 On the MRS console, click **Clusters** and choose **Active Clusters**. Click the target cluster name to access the cluster details page.

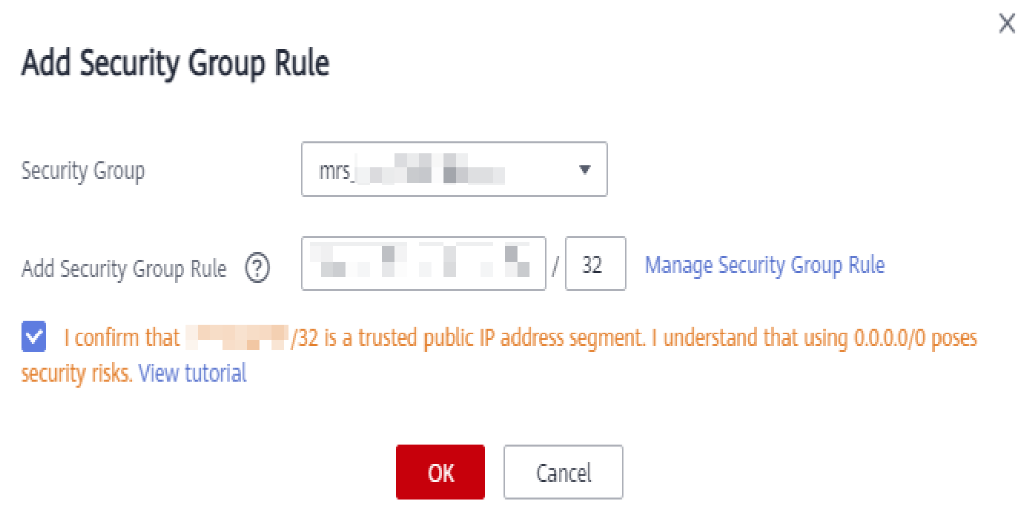
Step 2 Click **Add Security Group Rule** on the right of **EIP**. See **Figure 32-5**.

Figure 32-5 Cluster details



Step 3 On the **Add Security Group Rule** page, add the IP address segment for users to access the public network and select.**I confirm that the authorized object is a trusted public IP address range. Do not use 0.0.0.0/0. Otherwise, security risks may arise.** See **Figure 32-6**.

Figure 32-6 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 1** to **Step 4**. If you want to view, modify, or delete a security group rule, click **Manage Security Group Rule**.

Step 4 Click **OK**.

----End

32.2.2 Accessing FusionInsight Manager (MRS 3.*x* or Later)

Scenario

In MRS 3.*x* or later, FusionInsight Manager is used to monitor, configure, and manage clusters. After a cluster is installed, you can use an account to log in to FusionInsight Manager.

Currently, you can access FusionInsight Manager using the following methods:

- **Accessing FusionInsight Manager Using EIP**
- **Accessing FusionInsight Manager Using Direct Connect**
- **Accessing FusionInsight Manager from an ECS**

You can switch the access methods between **EIP** and **Direct Connect** on the MRS console by performing the following steps:

Log in to the MRS management console and click the desired cluster. On the displayed page, click ⇌ next to **MRS Manager** on the **Dashboard** tab, and switch the access method.

NOTE

If you cannot log in to the WebUI of the component, access FusionInsight Manager by referring to **Accessing FusionInsight Manager from an ECS**.

FusionInsight Manager cannot be accessed when the cluster is in any of the following states:

Starting, Stopping, Stopped, Deleting, Deleted, and Frozen.

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 **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.

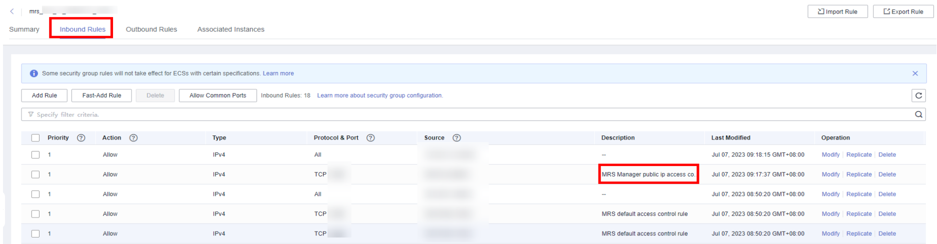
   NOTE

   - 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**.

NOTE

"MRS Manager public ip access control rule" is added to the **Description** column of the the security group rule you added. To view this description, choose **Manage Security Group Rule**, click **Security Group**, and click the **Inbound Rules** tab.



4. Select the information to be confirmed and click **OK**.

NOTE

Click on the right of **Access Manager** to change the FusionInsight Manager access mode. For details about how to access FusionInsight Manager by using **Direct Connect**, see **Accessing FusionInsight Manager by Using Direct Connect**.

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.

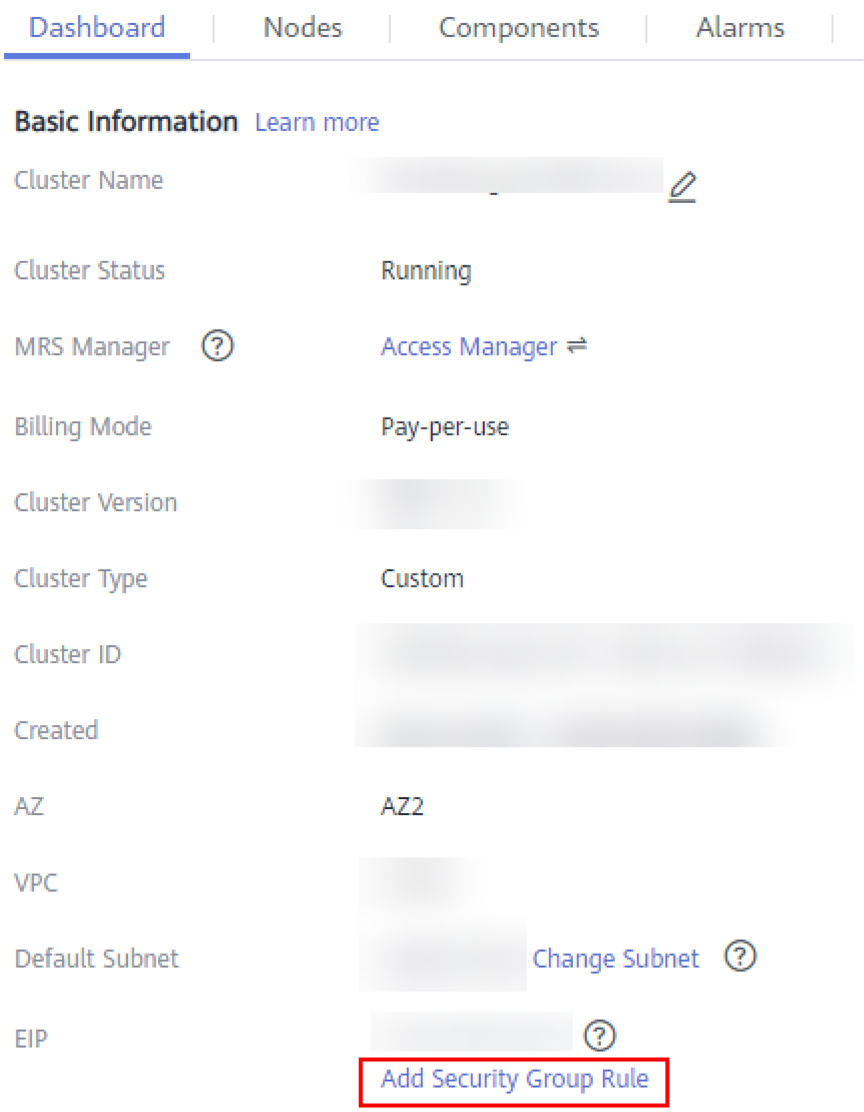
Step 6 On the MRS management console, choose **Clusters** > **Active Clusters**. Click the target cluster name to access the cluster details page.

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.

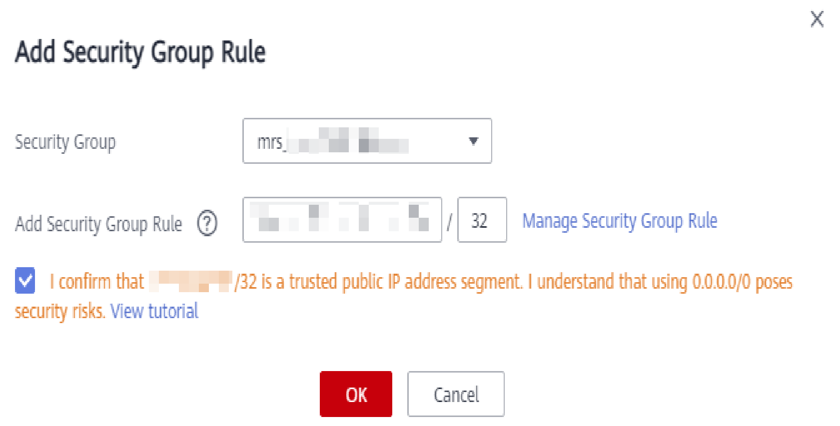
Step 7 Click **Add Security Group Rule** next to **EIP**.

Figure 32-7 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 32-8**.

Figure 32-8 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 by Using Direct Connect

You need to ensure that Direct Connect is available, and the connection between the local data center and the online VPC has been established.

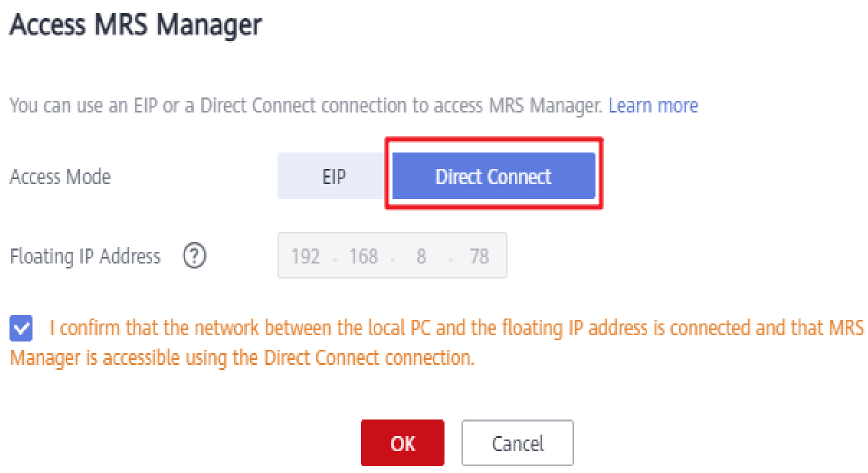
Step 1 Log in to the MRS console.

Step 2 Click the name of the cluster to enter its details page.

Step 3 On the **Dashboard** tab page of the cluster details page, click **Access Manager** next to **MRS Manager**.

Step 4 Set **Access Mode** to **Direct Connect** and select **I confirm that the network between the local PC and the floating IP address is connected and that MRS Manager is accessible using the Direct Connect connection**.

The floating IP address is automatically allocated by MRS to access MRS Manager. Before using Direct Connect to access MRS Manager, ensure that the connection between the local data center and the online VPC has been established.



Step 5 Click **OK**. The MRS Manager login page is displayed. Enter the username **admin** and the password set during cluster creation.

----End

Accessing FusionInsight Manager from an ECS

Step 1 Log in to the MRS management console.

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**.

If "Failed to add security group rules." is displayed, check whether the security group quota is sufficient. If more quotas are needed, increase the quotas or delete security group rules that are no longer used.

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 security group rules are required for logging in to the 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

32.3 Using an MRS Client

32.3.1 Installing a Client (MRS 3.*x* or Later)

Scenario

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**.

You can install the clients on a node in or outside the cluster.

After modifying the server configuration of a cluster component, reinstall the component client to ensure the server and the client both run the same version to provide services properly.

Prerequisites

- If the node where the client is to be installed is outside the cluster, the node must be able to communicate with the nodes in the cluster. Otherwise, client installation will fail.
- The node where the client is to be installed must have the NTP service enabled and synchronized time with the server. Otherwise, client installation will fail.
- You install the client as user **root** or any OS user. The user must have the operation permission on the client file storage directory and installation directory. The permission on the two directories is **755**.

  This section uses the OS user **user_client** as an example to describe how to install the client in the **/opt/hadoopclient** directory.
- When you install the client as a user other than **omm** or **root**, and the **/var/tmp/patch** directory already exists, you have changed the

permission for the directory to **777** and changed the permission for the logs in the directory to **666**.

Installing a Client on a Node Inside a Cluster

Step 1 Obtain the client software package.

Log in to FusionInsight Manager by referring to **Accessing FusionInsight Manager (MRS 3.x or Later)**. On the **Cluster** > **Dashboard** page, click the more sign (...) and select **Download Client**. In the **Download Cluster Client** dialog box displayed, configure parameters and click **OK**.

NOTE

- The client software package downloaded from the FusionInsight Manager homepage contains the clients of all services (excluding Flume) in the cluster. To download the client of a single service, choose **Cluster** > **Services** > *Service name*, click **More**, and select **Download Client**.
- For MRS 3.3.0 or later, click **Download Client** on the home page.

Table 32-1 Client download parameters

Parameter	Description	Example Value
Select Client Type	• **Complete Client**: contains the complete client software package and configuration files, which applies to non-development task scenarios. • **Configuration Files Only**: downloads only client configuration files in the scenario where the administrator modifies the server configuration on FusionInsight Manager after the complete client is downloaded and installed in an application development task, and developers need to update client configuration files.	Complete Client
Select Platform Type	**The client type must match the architecture of the node where the client is to be installed. Otherwise, the installation fails. For clusters of the LTS version, only the client software package whose type is the same as that of FusionInsight Manager can be downloaded.** • **x86_64**: indicates the client software package that can be deployed on a x86 platform. • **aarch64**: indicates the client software package that can be deployed on a TaiShan server.	x86_64

Parameter	Description	Example Value
Save to Path	The path for storing the client software package on the active OMS node ● Select **Save to Path**: Customize the path for storing the client software package on the active OMS node. User **omm** must have the read, write, and execute permissions on the path. If the path is not changed, the client file generated is saved in the **/tmp/FusionInsight-Client** directory on the active OMS node in the cluster by default. ● Not to select **Save to Path**: The generated client file is automatically downloaded and saved to the local host. Before installing the client, you need to upload the file to a specified directory on the target node.	Select **Save to Path**

Step 2 Copy the client software package to a specified directory on the node where the client is to be installed.

By default, the client software package is stored on the active OMS node in the cluster. To install the client on other nodes in the cluster, log in to the active OMS node as user **omm** and run the following command to copy the software package to the specified node. Otherwise, skip this step.

For example, copy the software package to the **/tmp/clienttemp** directory.

scp -p */tmp/FusionInsight-Client/***FusionInsight_Cluster_1_Services_Client.tar** *IP address of the node where the client is to be installed:/tmp/clienttemp*

Step 3 Log in to the target node as a user, for example, **user_client**.

NOTE

You can install the client as user **root** or any other OS user. The user must have the operation permission on the client file storage directory and installation directory. The permission on the two directories is **755**.

Step 4 Decompress the client software package.

1. Go to the directory where the package is stored, for example, **/tmp/clienttemp**.

   cd /tmp/clienttemp

2. Run the following commands to decompress the package and obtain **FusionInsight_Cluster_1_Services_ClientConfig.tar**:

   tar -xvf FusionInsight_Cluster_1_Services_Client.tar

3. 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
```

4. Run the following command to decompress **FusionInsight_Cluster_1_Services_ClientConfig.tar**:

   tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar

Step 5 Go to the package directory and run the following command to install the client to a specified directory:

cd FusionInsight_Cluster_1_Services_ClientConfig

./install.sh *Client installation directory*

For example, run the **./install.sh /opt/hadoopclient** command to install the client and wait until the installation is complete.

```
...
The component client is installed successfully
```

NOTE

- **If there is no specified client installation directory, it will be automatically created. If you specify an existing directory, it must be empty, and the path cannot contain spaces. The value can contain only uppercase letters, lowercase letters, digits, and underscores (_).**
- You need to manually delete the client installation directory when uninstalling a client.
- To ensure that the installed client can be used only by the installation user, add the **-o** parameter during the installation. For example, run the **./install.sh /opt/hadoopclient -o** command to install the client.

Step 6 Use the client by referring to "Using the client of Each Component".

----End

Installing a Client on a Node Outside a Cluster

Step 1 Create an ECS that meets the following requirements:

- A Linux ECS has been prepared. For details about the supported OS of the ECS, see **Table 32-2**.

Table 32-2 Reference list

CPU Architecture	OS	Supported Version
x86 computing	EulerOS	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)	EulerOS	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.

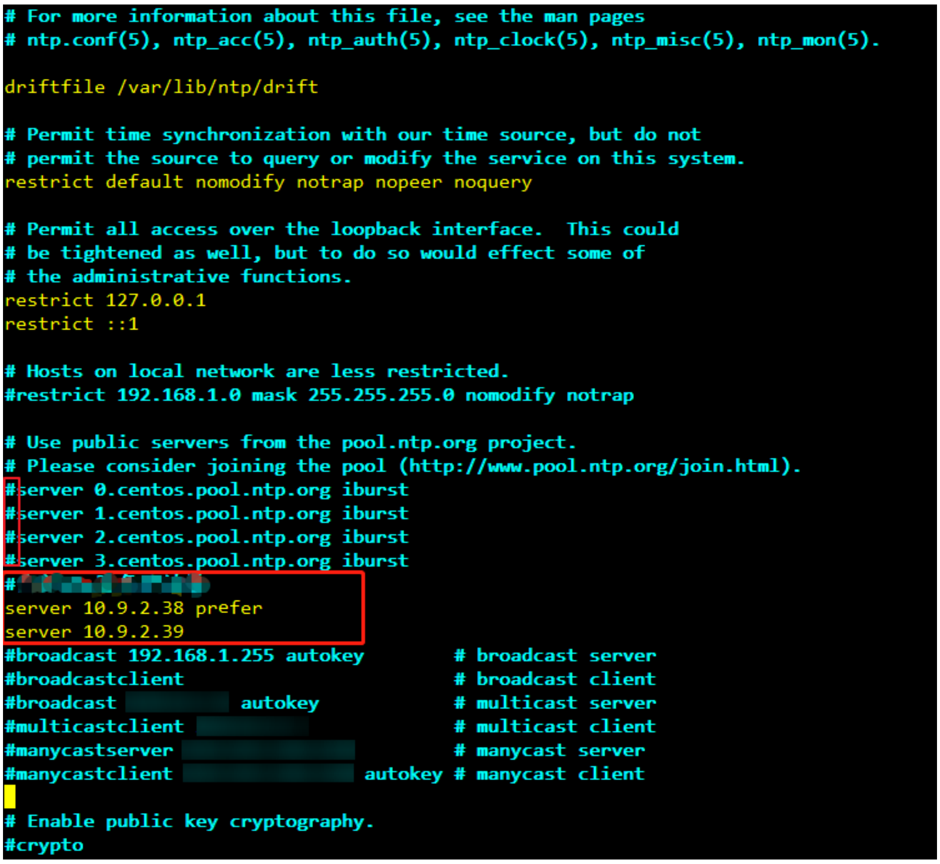
  If the NTP service is not installed, run the **yum install ntp -y** command to install it when the **yum** source is configured.
- A user can log in to the Linux ECS using the password (in SSH mode).
- All ports in the inbound direction of the MRS cluster security group are open to the client node. For details, see **Adding a Security Group Rule**.

Step 2 Perform NTP time synchronization to synchronize the time of nodes outside the cluster with the time of the MRS cluster.

1. 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 32-9 Adding the master node IP addresses



```
# For more information about this file, see the man pages
# ntp.conf(5), ntp_acc(5), ntp_auth(5), ntp_clock(5), ntp_misc(5), ntp_mon(5).

driftfile /var/lib/ntp/drift

# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default nomodify notrap nopeer noquery

# Permit all access over the loopback interface.  This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1
restrict ::1

# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
#server 0.centos.pool.ntp.org iburst
#server 1.centos.pool.ntp.org iburst
#server 2.centos.pool.ntp.org iburst
#server 3.centos.pool.ntp.org iburst
#[redacted]
server 10.9.2.38 prefer
server 10.9.2.39
#broadcast 192.168.1.255 autokey        # broadcast server
#broadcastclient                        # broadcast client
#broadcast [redacted] autokey           # multicast server
#multicastclient [redacted]             # multicast client
#manycastserver [redacted]              # manycast server
#manycastclient [redacted] autokey      # manycast client

# Enable public key cryptography.
#crypto
```

2. Run the **service ntpd stop** command to stop the NTP service.
3. 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.

4. Run the **service ntpd start** or **systemctl restart ntpd** command to start the NTP service.
5. Run the **ntpstat** command to check the time synchronization result.

Step 3 Obtain the client software package.

Log in to FusionInsight Manager by referring to **Accessing FusionInsight Manager (MRS 3.x or Later)**. On the **Cluster** > **Dashboard** page, click the more sign (...) and select **Download Client**. In the **Download Cluster Client** dialog box displayed, configure parameters and click **OK**.

NOTE

- The client software package downloaded from the FusionInsight Manager homepage contains the clients of all services (excluding Flume) in the cluster. To download the client of a single service, choose **Cluster** > **Services** > *Service name*, click **More**, and select **Download Client**.
- For MRS 3.3.0 or later, click **Download Client** on the home page.

Table 32-3 Client download parameters

Parameter	Description	Example Value
Select Client Type	• **Complete Client**: contains the complete client software package and configuration files, which applies to non-development task scenarios. • **Configuration Files Only**: downloads only client configuration files in the scenario where the administrator modifies the server configuration on FusionInsight Manager after the complete client is downloaded and installed in an application development task, and developers need to update client configuration files.	Complete Client
Select Platform Type	**The client type must match the architecture of the node where the client is to be installed. Otherwise, the installation fails. For clusters of the LTS version, only the client software package whose type is the same as that of FusionInsight Manager can be downloaded.** • **x86_64**: indicates the client software package that can be deployed on a x86 platform. • **aarch64**: indicates the client software package that can be deployed on a TaiShan server.	x86_64

Parameter	Description	Example Value
Save to Path	The path for storing the client software package on the active OMS node ● Select **Save to Path**: Customize the path for storing the client software package on the active OMS node. User **omm** must have the read, write, and execute permissions on the path. If the path is not changed, the client file generated is saved in the **/tmp/FusionInsight-Client** directory on the active OMS node in the cluster by default. ● Not to select **Save to Path**: The generated client file is automatically downloaded and saved to the local host. Before installing the client, you need to upload the file to a specified directory on the target node.	Select **Save to Path**

Step 4 Copy the client software package to a specified directory on the node where the client is to be installed.

The generated client software package is stored on the active OMS node of the cluster by default. You need to log in to the active OMS node as user **omm** and run the following command to copy the software package to a specified ECS:

For example, copy the software package to the **/tmp/clienttemp** directory.

scp -p */tmp/FusionInsight-Client/***FusionInsight_Cluster_1_Services_Client.tar** *IP address of the node where the client is to be installed:/tmp/clienttemp*

Step 5 Log in to the target node as a user, for example, **user_client**.

NOTE

You can install the client as user **root** or any other OS user. The user must have the operation permission on the client file storage directory and installation directory. The permission on the two directories is **755**.

Step 6 Decompress the client software package.

1. Go to the directory where the package is stored, for example, **/tmp/clienttemp**.

   cd /tmp/clienttemp

2. Run the following commands to decompress the package and obtain **FusionInsight_Cluster_1_Services_ClientConfig.tar**:

   tar -xvf FusionInsight_Cluster_1_Services_Client.tar

3. 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
   ```

4. Run the following command to decompress **FusionInsight_Cluster_1_Services_ClientConfig.tar**:

   tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar

Step 7 Check the network connection of the client.

1. Ensure that the host where the client is installed can communicate with the hosts listed in the **hosts** file stored in the directory containing the decompressed package, for example, **/tmp/FusionInsight_Cluster_1_Services_ClientConfig/hosts**.
2. If the host where the client is installed is not a host in the cluster, you need to set the mapping between the host name and the service plane IP address for each cluster node in **/etc/hosts**, as user **root**. Each host name uniquely maps an IP address. You can perform the following steps to import the domain name mapping of the cluster to the **hosts** file:
   a. Switch to user **root** or a user who has the permission to modify the **hosts** file.

      su - root

   b. Go to the directory where the client package is decompressed.

      cd /tmp/clienttemp/FusionInsight_Cluster_1_Services_ClientConfig

   c. Run the **cat realm.ini >> /etc/hosts** command to import the domain name mapping to the **hosts** file.

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.
- If Spark tasks are executed in yarn-client mode, add the **spark.driver.host** parameter to the file *Client installation directory***/Spark/spark/conf/spark-defaults.conf** and set the parameter to the client IP address.
- If the yarn-client mode is used, you need to configure the mapping between the IP address and host name of the client in the **hosts** file on the active and standby Yarn nodes (ResourceManager nodes in the cluster) to make sure that the Spark web UI is properly displayed.

Step 8 Log in to the node where the client is to be installed as user **user_client**, go to the client software package directory, and run the following command to install the client to a specified directory:

cd /tmp/clienttemp/FusionInsight_Cluster_1_Services_ClientConfig

./install.sh *Client installation directory*

For example, run the **./install.sh /opt/hadoopclient** command to install the client and wait until the installation is complete.

```
...
The component client is installed successfully
```

NOTE

- **If there is no specified client installation directory, it will be automatically created. If you specify an existing directory, it must be empty, and the path cannot contain spaces. The value can contain only uppercase letters, lowercase letters, digits, and underscores (_).**
- You need to manually delete the client installation directory when uninstalling a client.
- To ensure that the installed client can be used only by the installation user, add the **-o** parameter during the installation. For example, run the **./install.sh /opt/hadoopclient -o** command to install the client.

Step 9 Use the client by referring to "Using the client of Each Component".

----End

32.3.2 Installing a Client (Versions Earlier Than 3.x)

Scenario

An MRS client is required. The MRS cluster client can be installed on the Master or Core node in the cluster or on a node outside the cluster.

After a cluster of versions earlier than MRS 3.x is created, a client is installed on the active Master node by default. You can directly use the client. The installation directory is **/opt/client**.

For details about how to install a client of MRS 3.x or later, see **Installing a Client (MRS 3.x or Later)**.

NOTE

If a client has been installed on the node outside the MRS cluster and the client only needs to be updated, update the client using the user who installed the client, for example, user **root**.

Installing a Client on the Core Node

1. Log in to MRS Manager and choose **Services** > **Download Client** to download the client installation package to the active management node.

   NOTE

   If only the client configuration file needs to be updated, see method 2 in **Updating a Client (Versions Earlier Than 3.x)**.
2. Use the IP address to search for the active management node, and log in to the active management node using VNC.
3. Log in to the active management node, and run the following command to switch the user:

   sudo su - omm
4. On the MRS management console, view the IP address on the **Nodes** tab page of the specified cluster.

   Record the IP address of the Core node where the client is to be used.
5. On the active management node, run the following command to copy the client installation package to the Core node:

scp -p /tmp/MRS-client/MRS_Services_Client.tar *IP address of the Core node*:**/opt/client**

6. Log in to the Core node as user **root**.

   Master nodes support Cloud-Init. The preset username for Cloud-Init is **root** and the password is the one you set during cluster creation.

7. Run the following commands to install the client:

   cd /opt/client

   tar -xvf MRS_Services_Client.tar

   tar -xvf MRS_Services_ClientConfig.tar

   cd /opt/client/MRS_Services_ClientConfig

   ./install.sh *Client installation directory*

   For example, run the following command:

   ./install.sh /opt/client

8. For details about how to use the client, see **Using an MRS Client**.

Using an MRS 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 the Cluster

Step 1 Create an ECS that meets the following requirements:

- For clusters of versions earlier than MRS 3.*x*, you need to confirm the CPU architecture of the current MRS cluster nodes. For clusters of versions earlier than MRS 3.*x*, the CPU architecture of the ECS must be the same as that of the MRS cluster node. For MRS 3.*x* or later, the MRS client is compatible with both of the following CPU architectures.
- An ECS has been prepared. For details about the OS and its version of the ECS, see **Table 32-4**.

Table 32-4 Reference list

CPU Architecture	OS	Supported Version
x86 computing	EulerOS	– Available: EulerOS 2.2 – Available: EulerOS 2.3 – Available: EulerOS 2.5

For example, a user can select an ECS running the EulerOS.

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 is the same as that of the master node in the MRS cluster.

  If this requirement is not met, modify the ECS security group or configure the inbound and outbound rules of the ECS security group to allow the ECS security group to be accessed by all security groups of MRS cluster nodes.
- To enable users to log in to a Linux ECS using a password (SSH), see *"Instances" > "Logging In to a Linux ECS" > "Login Using an SSH Password" in Elastic Cloud Server User Guide*.
- 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**.

Step 2 Log in to MRS Manager. For details, see **Accessing MRS Manager (Versions Earlier Than MRS 3.x)**. Then, choose **Services**.

Step 3 Click **Download Client**.

Step 4 In **Client Type**, select **All client files**.

Step 5 In **Download To**, select **Remote host**.

Step 6 Set **Host IP Address** to the IP address of the ECS, **Host Port** to **22**, and **Save Path** to **/tmp**.

- If the default port **22** for logging in to an ECS using SSH has been changed, set **Host Port** to the new port.
- **Save Path** contains a maximum of 256 characters.

Step 7 Set **Login User** to **root**.

If other users are used, ensure that the users have read, write, and execute permission on the save path.

Step 8 Select **Password** or **SSH Private Key** for **Login Mode**.

- **Password**: Enter the password of user **root** set during cluster creation.
- **SSH Private Key**: Select and upload the key file used for creating the cluster.

Step 9 Click **OK** to generate a client file.

If the following information is displayed, the client package is saved. Click **Close**. Obtain the client file from the save path on the remote host that is set when the client is downloaded.

```
Client files downloaded to the remote host successfully.
```

If the following information is displayed, check the username, password, and security group configurations of the remote host. Ensure that the username and password are correct and an inbound rule of the SSH (22) port has been added to the security group of the remote host. And then, go to **Step 2** to download the client again.

```
Failed to connect to the server. Please check the network connection or parameter settings.
```

NOTE

Generating a client will occupy a large number of disk I/Os. You are advised not to download a client when the cluster is being installed, started, and patched, or in other unstable states.

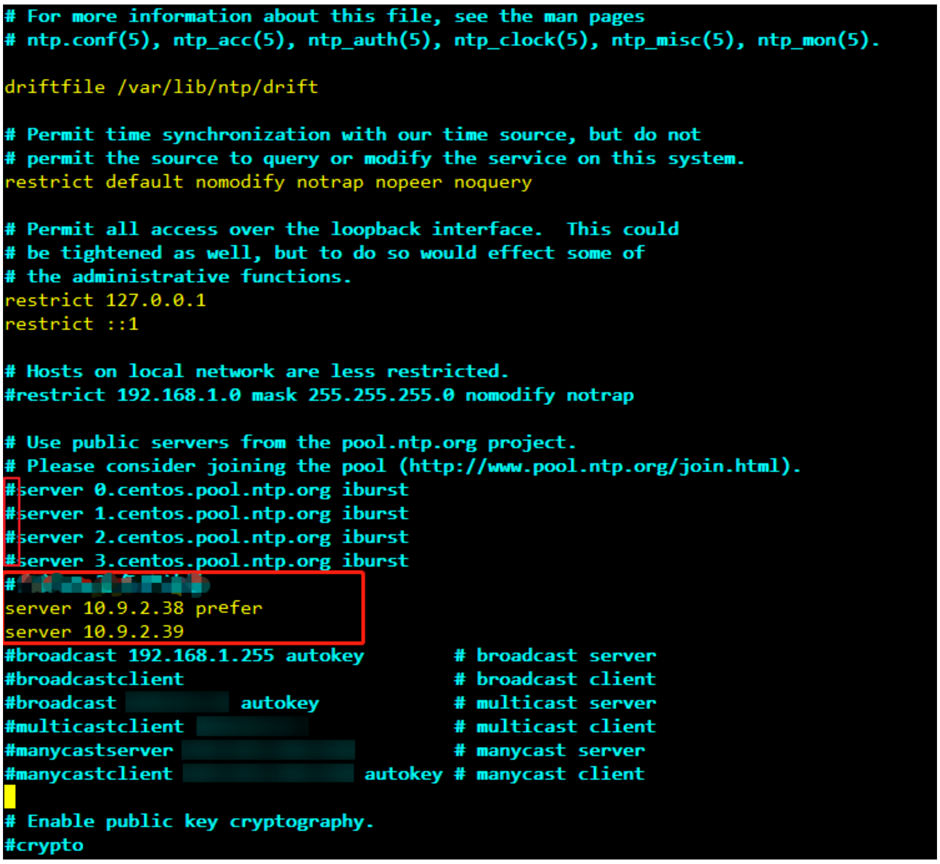
Step 10 Log in to the ECS using VNC. For details, see **Instance** > **Logging In to a Linux** > **Logging In to a Linux** in the *Elastic Cloud Server User Guide*

All images support Cloud-Init. The preset username for Cloud-Init is **root** and the password is the one you set during cluster creation. It is recommended that you change the password upon the first login.

Step 11 Perform NTP time synchronization to synchronize the time of nodes outside the cluster with the time of the MRS cluster.

1. Check whether the NTP service is installed. If it is not installed, run the **yum install ntp -y** command to install it.
2. Run the **vim /etc/ntp.conf** command to edit the NTP client configuration file, add the IP address of the Master node in the MRS cluster, and comment out the IP addresses of other servers.
```
server master1_ip prefer
server master2_ip
```

Figure 32-10 Adding the master node IP addresses



3. Run the **service ntpd stop** command to stop the NTP service.
4. 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.
5. Run the **service ntpd start** or **systemctl restart ntpd** command to start the NTP service.
6. Run the **ntpstat** command to check the time synchronization result:

Step 12 On the ECS, switch to user **root** and copy the installation package in **Save Path** in **Step 6** to the **/opt** directory. For example, if **Save Path** is set to **/tmp**, run the following commands:

sudo su - root

cp /tmp/MRS_Services_Client.tar /opt

Step 13 Run the following command in the **/opt** directory to decompress the package and obtain the verification file and the configuration package of the client:

tar -xvf MRS_Services_Client.tar

Step 14 Run the following command to verify the configuration file package of the client:

sha256sum -c MRS_Services_ClientConfig.tar.sha256

The command output is as follows:

```
MRS_Services_ClientConfig.tar: OK
```

Step 15 Run the following command to decompress **MRS_Services_ClientConfig.tar**:

tar -xvf MRS_Services_ClientConfig.tar

Step 16 Run the following command to install the client to a new directory, for example, **/opt/Bigdata/client**. A directory is automatically generated during the client installation.

sh /opt/MRS_Services_ClientConfig/install.sh /opt/Bigdata/client

If the following information is displayed, the client has been successfully installed:

```
Components client installation is complete.
```

Step 17 Check whether the IP address of the ECS node is connected to the IP address of the cluster Master node.

For example, run the following command: **ping** *Master node IP address*.

- If yes, go to **Step 18**.
- If no, check whether the VPC and security group are correct and whether the ECS and the MRS cluster are in the same VPC and security group, and go to **Step 18**.

Step 18 Run the following command to configure environment variables:

source /opt/Bigdata/client/bigdata_env

Step 19 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**

Step 20 Run the client command of a component.

For example, run the following command to query the HDFS directory:

hdfs dfs -ls /

----End

32.3.3 Updating a Client (Version 3.x or Later)

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.

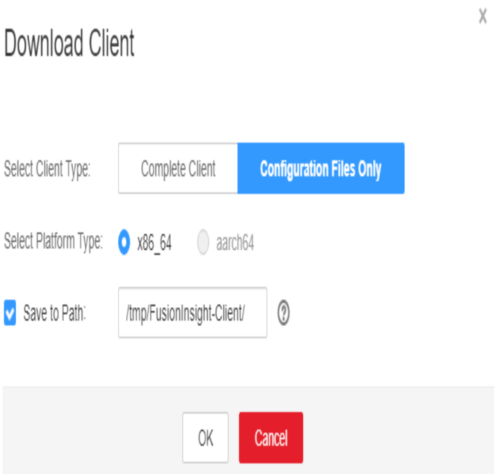
Updating the Client Configuration

Method 1:

Step 1 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager (MRS 3.x or Later)**. 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 OMS.

NOTE

To obtain the floating IP address of OMS, log in to the Master2 node remotely, and run the **ifconfig** command. In the command output, **eth0:wsom** indicates the floating IP address of OMS. Record the value of **inet**. If the floating IP address of OMS cannot be queried on the Master2 node, switch to the Master1 node to query and record the floating IP address. If there is only one Master node, query and record the IP address on the Master node.

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

32.3.4 Updating a Client (Versions Earlier Than 3.x)

NOTE

This section applies to clusters of versions earlier than MRS 3.x. For MRS 3.x or later, see **Updating a Client (Version 3.x or Later)**.

Updating a Client Configuration File

Scenario

An MRS cluster provides a client for you to connect to a server, view task results, or manage data. Before using an MRS client, you need to download and update the client configuration file if service configuration parameters are modified and a service is restarted on MRS Manager.

During cluster creation, the original client is stored in the **/opt/client** directory on all nodes in the cluster by default. After the cluster is created, only the client of a Master node can be directly used. To use the client of a Core node, you need to update the client configuration file first.

Procedure

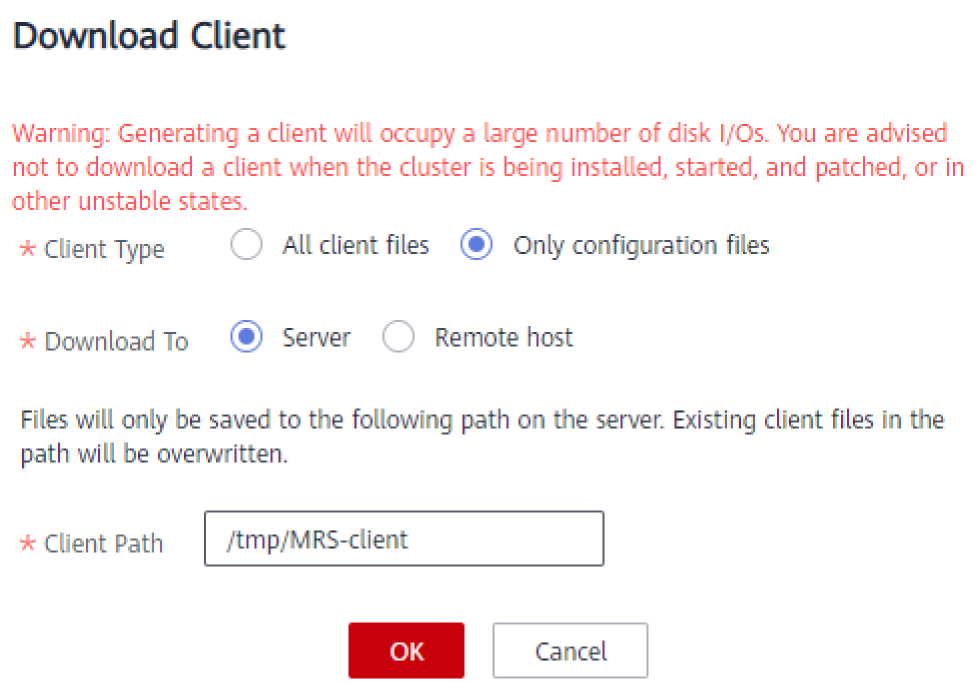
Method 1: applicable to all versions

Step 1 Log in to MRS Manager. For details, see **Accessing MRS Manager (Versions Earlier Than MRS 3.x)**. Then, choose **Services**.

Step 2 Click **Download Client**.

Set **Client Type** to **Only configuration files**, **Download To** to **Server**, and click **OK** to generate the client configuration file. The generated file is saved in the **/tmp/MRS-client** directory on the active management node by default. You can customize the file path.

Figure 32-11 Downloading the client configuration file



Step 3 Query and log in to the active Master node.

Step 4 If you use the client in the cluster, run the following command to switch to user **omm**. If you use the client outside the cluster, switch to user **root**.

sudo su - omm

Step 5 Run the following command to go to the client directory:

cd *Client installation directory*

Step 6 Run the following command to update client configurations:

sh refreshConfig.sh *Client installation directory Full path of the client configuration file package*

For example, run the following command:

sh refreshConfig.sh /opt/Bigdata/client /tmp/MRS-client/MRS_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.
```

----End

Method 2:

Step 1 After the cluster is installed, run the following command to switch to user **omm**. If you use the client outside the cluster, switch to user **root**.

sudo su - omm

Step 2 Run the following command to go to the client directory:

cd *Client installation directory*

Step 3 Run the following command and enter the name of an MRS Manager user with the download permission and its password (for example, the username is **admin** and the password is the one set during cluster creation) as prompted to update client configurations.

sh autoRefreshConfig.sh

Step 4 After the command is executed, the following information is displayed, where *XXX* indicates the name of the component installed in the cluster. To update client configurations of all components, press **Enter**. To update client configurations of some components, enter the component names and separate them with commas (,).

```
Components "xxx" have been installed in the cluster. Please input the comma-separated names of the
components for which you want to update client configurations. If you press Enter without inputting any
component name, the client configurations of all components will be updated:
```

If the following information is displayed, the configurations have been updated successfully.

```
Succeed to refresh components client config.
```

If the following information is displayed, the username or password is incorrect.

```
login manager failed,Incorrect username or password.
```

NOTE

- This script automatically connects to the cluster and invokes the **refreshConfig.sh** script to download and update the client configuration file.
- By default, the client uses the floating IP address specified by **wsom=xxx** in the **Version** file in the installation directory to update the client configurations. To update the configuration file of another cluster, modify the value of **wsom=xxx** in the **Version** file to the floating IP address of the corresponding cluster before performing this step.

----End

Fully Updating the Original Client of the Active Master Node

Scenario

During cluster creation, the original client is stored in the **/opt/client** directory on all nodes in the cluster by default. The following uses **/opt/Bigdata/client** as an example.

- For a normal MRS cluster, you will use the pre-installed client on a Master node to submit a job on the management console page.
- You can also use the pre-installed client on the Master node to connect to a server, view task results, and manage data.

After installing the patch on the cluster, you need to update the client on the Master node to ensure that the functions of the built-in client are available.

Procedure

Step 1 Log in to MRS Manager. For details, see **Accessing MRS Manager (Versions Earlier Than MRS 3.x)**. Then, choose **Services**.

Step 2 Click **Download Client**.

Set **Client Type** to **All client files**, **Download To** to **Server**, and click **OK** to generate the client configuration file. The generated file is saved in the **/tmp/MRS-client** directory on the active management node by default. You can customize the file path.

Step 3 Query and log in to the active Master node.

Step 4 On the ECS, switch to user **root** and copy the installation package to the **/opt** directory.

sudo su - root

cp /tmp/MRS-client/MRS_Services_Client.tar /opt

Step 5 Run the following command in the **/opt** directory to decompress the package and obtain the verification file and the configuration package of the client:

tar -xvf MRS_Services_Client.tar

Step 6 Run the following command to verify the configuration file package of the client:

sha256sum -c MRS_Services_ClientConfig.tar.sha256

The command output is as follows:

```
MRS_Services_ClientConfig.tar: OK
```

Step 7 Run the following command to decompress **MRS_Services_ClientConfig.tar**:

tar -xvf MRS_Services_ClientConfig.tar

Step 8 Run the following command to move the original client to the **/opt/Bigdata/client_bak** directory:

mv /opt/Bigdata/client /opt/Bigdata/client_bak

Step 9 Run the following command to install the client in a new directory. The client path must be **/opt/Bigdata/client**.

sh /opt/MRS_Services_ClientConfig/install.sh /opt/Bigdata/client

If the following information is displayed, the client has been successfully installed:

```
Components client installation is complete.
```

Step 10 Run the following command to modify the user and user group of the **/opt/Bigdata/client** directory:

chown omm:wheel /opt/Bigdata/client -R

Step 11 Run the following command to configure environment variables:

source /opt/Bigdata/client/bigdata_env

Step 12 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**

Step 13 Run the client command of a component.

For example, run the following command to query the HDFS directory:

hdfs dfs -ls /

----End

Fully Updating the Original Client of the Standby Master Node

Step 1 Repeat **Step 1** to **Step 3** to log in to the standby Master node, and run the following command to switch to user **omm**:

sudo su - omm

Step 2 Run the following command on the standby master node to copy the downloaded client package from the active master node:

scp omm@*master1 nodeIP address***:/tmp/MRS-client/MRS_Services_Client.tar /tmp/MRS-client/**

NOTE

- In this command, **master1** node is the active master node.
- **/tmp/MRS-client/** is an example target directory of the standby master node.

Step 3 Repeat **Step 4** to **Step 13** to update the client of the standby Master node.

----End